ASIC Physical Implementation Methodology for Intel Advanced X86 CPU Design

Project report submitted in partial fulfilment of the requirement for the Award of the Degree of

MASTER OF TECHNOLOGY

in VLSI DESIGN

Submitted By

Vishesh Chaudhary

6024620029

Under Supervision of

Dr. Gitanjli Chandwani Manocha

(Assistant Professor)

&

Dr. Hem Dutt Joshi

(Professor & Associate Dean)

&

Sivaraman P

(Technical Manager at Intel)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

JANUARY-JUNE, 2026

DECLARATION

I, **Vishesh Chaudhary** hereby declare that the work presented in this thesis entitled **"ASIC Physical Implementation Methodology for Intel Advanced X86 CPU Design"** in partial fulfillment of the requirement for the award of degree of **Master of Technology (VLSI Design**) submitted at **Department of Electronics and Communication Engineering**, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala is an authentic record of work carried out under supervision of **Dr. Gitanjali Chandwani Manocha (Assistant Professor, ECED, Thapar Institute of Engineering & Technology)** and **Dr. Hem Dutt Joshi (Professor & Associate Dean(DCT), ECED, Thapar Institute of Engineering & Technology)** from **January 2026** to **June 2026**. The matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Sign Vishesh

Date: 30th June 2026

Vishesh Chaudhary

(6024620029)

Sivaraman P. Technical Manager Intel Technology India Pvt. Ltd., Bangalore, Karnataka	Dr. Gitanjali Chandwani Manocha Assistant Professor Department of Electronics and Communication Engineering, Thapar Institute of Engineering & Technology, Patiala
	Dr. Hem Dutt Joshi Professor & Associate Dean (DCT) Department of Electronics and Communication Engineering, Thapar Institute of Engineering & Technology, Patiala

CERTIFICATE

Tel: +91-80-6211 3000
Fax: +91-80-6280 9010

To Whomsoever It May Concern

WWID: 12304899

Employee Name: Vishesh Chaudhary

Internship Dates: 06/16/2025 to 06/12/2026

The letter is to confirm the mentioned above has undergone internship at Intel Technology India Private Limited. We wish you all the best for your future assignments.

Yours Sincerely,
For Intel Technology India Pvt. Ltd.

Harishkumar S
Authorized Signatory

Intel Technology India Pvt. Ltd.

Registered Office:
Intel Technology India Pvt Ltd.
#23-56P, Outer Ring Road, Devarabeesanahalli, Varthur Hobli, Bellandur Post
Bengaluru 560 103, Karnataka, India
CIN – U85110KA1997PTC021606

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the individuals who have played an instrumental role in my internship journey at Intel. Their guidance, support, and encouragement have been invaluable, and I am truly grateful for their contributions.

First and foremost, I would like to extend my sincere thanks to my manager, Sivaraman P. His wealth of knowledge, expertise, and guidance have been instrumental in shaping my professional growth during this internship. His patience, encouragement, and willingness to share his insights and experiences have been truly inspiring. I am grateful for the time and effort he dedicated to mentoring me and for his unwavering support throughout the internship.

I would like to express my gratitude to my college mentors, Dr. Gitanjli Chandwani Manocha and Dr. Hem Dutt Joshi, for their continuous support and guidance throughout my internship. Advice, wisdom, and encouragement have been invaluable in helping me navigate through the challenges and make the most out of this experience. I am thankful for their unwavering support and for being a constant source of motivation, their guidance, support, and encouragement have been invaluable, and I am truly grateful for their contributions.

I am deeply grateful to Dr. Kulbir Singh, Head of the Department (ECE), Thapar Institute of Engineering & Technology, Patiala, for providing me with a learning Environment and Infrastructure in ECED. I will be ignorant if I do not express my gratitude to the author of the references and other literature cited in this seminar.

ABSTRACT

The physical design phase in VLSI implementation is a critical stage that transforms a synthesized gate-level netlist into a layout suitable for silicon fabrication. This process involves several sub-stages, including floor-planning, placement, clock tree synthesis (CTS), routing, and signoff, each of which must be meticulously executed to meet the design's performance, power, and area (PPA) objectives. Advanced Electronic Design Automation (EDA) tools play a central role in this flow, employing algorithms for congestion-aware placement, timing-driven routing, and multi-mode multi-corner (MMMC) analysis. These techniques ensure efficient cell distribution, accurate interconnect optimization, and robust validation across varying process, voltage, and temperature (PVT) conditions.

Power optimization is equally vital in achieving design efficiency. The use of Unified Power Format (UPF) enables early specification of power domains, voltage islands, and power gating strategies, allowing synthesis and physical design tools to automatically insert level shifters, isolation cells, and retention registers. Unified toolchains such as Cadence Innovus and Synopsys Fusion Compiler offer comprehensive support for RTL-to-GDSII implementation, integrating all stages within a single environment. This seamless integration facilitates cross-stage feedback and iterative refinement of timing, power, and area metrics. Furthermore, optimization engines embedded in these platforms enhance design convergence and enable rapid achievement of PPA targets, making them indispensable in modern semiconductor design flows.

In addition to these conventional methodologies, this work also incorporated timing-driven Non-Default Rule (NDR) analysis to evaluate its impact on clock distribution and timing closure. By experimenting with different trunk and leaf configurations, measurable improvements in setup and hold margins were achieved, validating NDR as a powerful optimization technique that complements standard routing and ECO strategies. The inclusion of this analysis provides deeper insight into advanced timing optimization practices, reinforcing the importance of specialized routing rules in achieving robust design closure.

TABLE OF CONTENTS

Sr. No.	Name of Chapter	Page No.
	Declaration	
	Certificate	
	Acknowledgement	
	Abstract	i
	List of Figures	iv
	List of Tables	vi
Chapter 1	Introduction	1
1.1	ASIC Physical Design Flow in VLSI	2
1.2	Purpose of the Automated Placement and Routing (PnR) Flow	4
1.3	Advanced EDA Tools for VLSI Physical Design and ASIC Implementation	5
1.3.1	Cadence Genus	5
1.3.2	Cadence Innovus	6
1.3.3	Synopsys EDA Suite	6
1.3.4	Synopsys Prime Time	6
Chapter 2	Objective	7
Chapter 3	Literature Review	8
Chapter 4	Synthesis	18
4.1	Synthesis Flow	18
4.2	Input files required for Synthesis	20
4.2.1	Import Design	24
4.2.2	Unified Power file	24
4.2.3	Initialize Floorplan	25
4.2.4	Read Timing Constraints	26
4.2.4.1	Multi-Corner Multi-Mode	27
4.3	Output files generated after Synthesis	29
Chapter 5	Optimization	31
5.1	Static Timing Analysis	31
5.2	Setup Time	32

5.2.1 Setup Violation..... 32
5.3 Hold Time..... 35
5.3.1 Fixing Max Transition Violations..... 35
5.3.2 Fixing Max CapacitanceViolations..... 36
5.3.3 Fixing Crosstalk Violations..... 36
Chapter 6 Timing-Driven NDR Analysis..... 38
6.1 Introduction to NDR..... 38
6.2 Effects of NDR..... 38
6.3 Purpose of NDR..... 38
6.4 Rules of NDR..... 38
6.5 Overcoming Challenges of NDR..... 49
6.6 Timing Analysis..... 40
6.6.1 Default NDR (2w2s) Analysis..... 40
6.6.2 Trunk 1.5w2.5s and Leaf 2w2s Analysis..... 42
6.6.3 Trunk 1.5w2.5s and Leaf 1.5w2.5s Analysis..... 44
6.7 Conclusion Summary..... 46
Chapter 7 Timing Fixing with ECO..... 47
7.1 Introduction to ECO..... 47
7.2 Architectural Context within the Physical Design Flow..... 47
7.3 Comprehensive Breakdown of the Iterative ECO Cycle..... 47
7.4 The Loop Closure Mechanism..... 50
7.5 Detecting Timing Issues..... 51
7.6 Evaluating Timing Issues..... 51
7.7 Formulating ECO Issues..... 51
7.8 Applying ECO Adjustments..... 54
7.9 Validating ECO Adjustments..... 55
7.10 Engineering Best Practices..... 55
Chapter 8 Results..... 57
Chapter 9 Conclusion and Future Scope..... 59
9.1 Conclusion..... 59
9.2 Future Scope..... 59
References..... 61
Plagiarism Report..... 63

LISTS OF FIGURES

Sr. No.	Figure Details	Page No.
Figure 1.1	ASIC Design Flow.....	2
Figure 1.2	Power distribution network.....	3
Figure 1.3	Placement of IO pads, Standard cells and Macros hierarchy wise....	4
Figure 3.1	Example of efficient floor plan….....	10
Figure 3.2	ClockMesh….....	11
Figure 3.3	An illustration of proposed split-and-parallel physical design flow...	12
Figure 4.1	Synthesis Flow….....	20
Figure 4.2	Input files for synthesis….....	20
Figure 4.3	Propagation delay of an Inverter….....	21
Figure 4.4	Physical libraries file (.lef) ….....	22
Figure 5.1	Types of Logical path….....	31
Figure 5.2	Buffer Insertion with Equal Driving Strength….....	33
Figure 5.3	Flowchart for Setup Fix….....	33
Figure 5.4	Bounded Logic….....	34
Figure 5.5	Unbounded Logic.....	34
Figure 5.6	Splitting the fanout.....	36
Figure 6.1	2w2s NDR Configuration.....	41
Figure 6.2	Default NDR (2w2s) Layout.....	41
Figure 6.3	Snapshot of Innovus Timing Summary(2w2s)	42
Figure 6.4	Snapshot of PrimeTime Timing Summary(2w2s)	42
Figure 6.5	Trunk 1.5w2.5s and Leaf 2w2s Configuration.....	43
Figure 6.6	Snapshot of Innovus Timing Summary (Trunk 1.5w2.5s and Leaf 2w2s)	43
Figure 6.7	Snapshot of PrimeTime Timing Summary (Trunk 1.5w2.5s and Leaf 2w2s)	44
Figure 6.8	Trunk 1.5w2.5s and Leaf 1.5w2.5s Configuration.....	44
Figure 6.9	1.5w and 2.5s NDR Layout.....	45
Figure 6.10	Snapshot of Innovus Timing Summary (Trunk 1.5w2.5s and Leaf 1.5w2.5s)	45

Figure 6.11 Snapshot of PrimeTime Timing Summary (Trunk 1.5w2.5s and Leaf 1.5w2.5s) 46
Figure 7.1 Original Buffer Sizing before ECO..... 51
Figure 7.2 Buffer Upsizing after ECO for Timing Improvement..... 51
Figure 7.3 Single-Buffer Path with Long Signal Propagation Distance..... 52
Figure 7.4 Buffer Insertion with Equal Driving Strengths..... 52
Figure 7.5 Buffer Insertion Using Different Driving Strengths..... 53
Figure 7.6 Direct Source-to-Sink Connection after Buffer Removal..... 53
Figure 7.7 Single Buffer Driving Multiple Fanouts..... 53
Figure 7.8 Logic Duplication with Separate Buffers for Each Fanout..... 54

LISTS OF TABLES

Sr. No.	Table Details	Page No.
Table 1	Path Types and Description.....	32
Table 2	Setup Timings before ECO.....	57
Table 3	Setup Timings after ECO.....	57
Table 4	Hold Timing before ECO.....	58
Table 5	Hold Timing after ECO.....	58

CHAPTER 1

INTRODUCTION

Today, Very Large-Scale Integration (VLSI) makes it possible for a single chip to contain billions of transistors. This achievement is the outcome of many years of progress in materials science, manufacturing methods, and chip design, all of which made possible the design of devices that are smaller, faster, and more power-efficient than before possible. VLSI technology allows complex function to be fit into very small silicon areas, making integrated circuit design. It powers a wide variety of devices from smartphones and laptops to critical systems such as commercial aircraft, automobiles, and industrial automation. By incorporating analog, digital, memory, and mixed-signal components on a single chip led to the emergence of System-on-Chip (SoC) designs, which are now widely used in connected devices. New challenges have been introduced due to nano-sized transistors, such as increased leakage currents, device variability, and delays.

These issues led to adapt advanced transistor structures like as Fin-FETs and Gate-All-Around (GAA) FETs, and also innovations in low-power design techniques such as multi-threshold voltage (multi-Vt), power gating, and dynamic voltage and frequency scaling. The VLSI design flow is a process to convert a chip from high-level design to something that can be fabricated. It is divided into two halves. The front-end handles RTL, functional verification, and logic synthesis, making sure the design work as intended. The back end handles the next phase from there, dealing with the layout elements of the design: floorplanning, placement, clock tree synthesis, routing, and final signoff checks before it is ready for tape in.

Along with technical progress, VLSI has also opened the door to new computing approaches such as parallel processing, machine learning accelerators, and neuromorphic architectures. The communication between hardware and software design is supported by hardware description languages (HDLs) and high-level synthesis (HLS) which had made it easier to develop complex chip designs in the semiconductor industry. As the semiconductor companies continues to push to achieve higher performance, but its main goal is to improve efficiency, and greater levels of integration. It also plays a major role in supporting emerging technologies such as artificial intelligence, 5G, quantum computing, and edge devices, which shows how important it is for the future of electronics.

1.1 ASIC Physical Design Flow in VLSI

Once a design can become a chip, the RTL first need to be synthesized into a gate-level netlist. This is where we can find the physical design pipeline comes into picture. At this point, the physical design engineers take that gate-level netlist and work step-by-step like floorplanning, power planning, placement, clock tree synthesis, routing transforming it into a real layout while closely analysing area, power, and performance of the design where PPA is a classical trade-off. However, every design must follow the foundry's guidelines because even minor defects can lead to a silicon failure, thus prevent a chip from being manufactured.

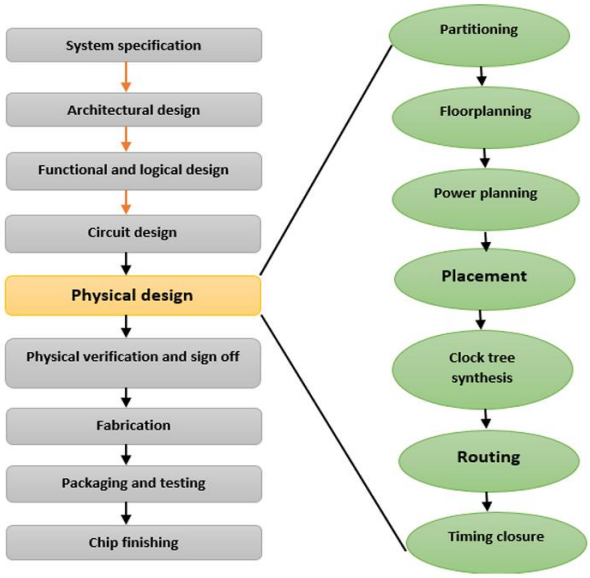

Fig. 1.1: ASIC Physical Design Flow

The major stages of the ASIC physical design flow are described below:

i. Partitioning: It is the initial step in the physical design flow where a large and complex ASIC design is divided into smaller, manageable blocks or modules. This hierarchical approach simplifies implementation, improves scalability, and allows for parallel development and reuse of IP blocks.

ii. Floor-planning: Floor-planning is the first step in physical design, where the overall layout of the chip is decided in terms of die size, core area, and aspect ratio. In this stage, macros, standard cells, and I/O pads are placed, and power domains and block boundaries are also defined. A good floorplan helps reduce routing congestion and makes it easier to achieve timing closure later in the design process.

iii. Power Planning: Power planning is carried out to ensure that the chip receives a stable and reliable power supply through an effective power distribution network (PDN). This stage involves creating power rings, stripes, and vias to properly distribute VDD and VSS throughout the design. It is an important step because it helps reduce problems like IR drop and electromigration, which can become serious in high-density chip designs.

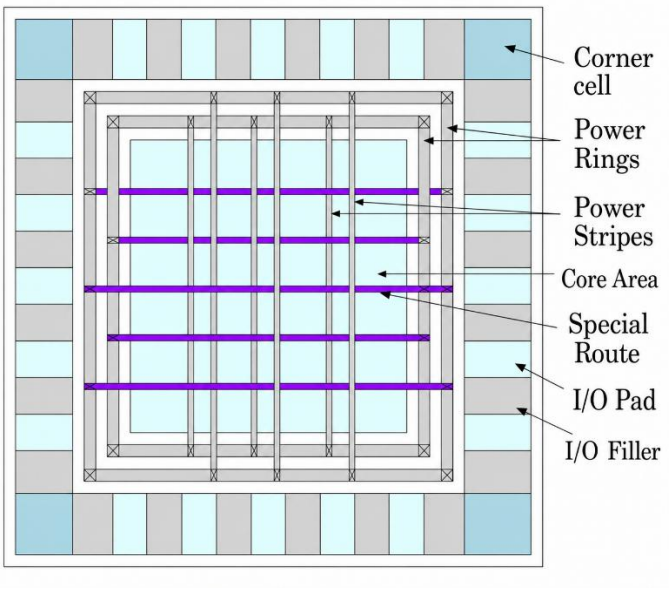

Fig.1.2: Power distribution network

iv. Placement: In this stage, standard cells are placed within the defined floorplan. Placement algorithms aim to minimize wirelength, reduce congestion, and optimize timing paths. Legalization is performed to align cells to placement grids and resolve overlaps, ensuring manufacturability.

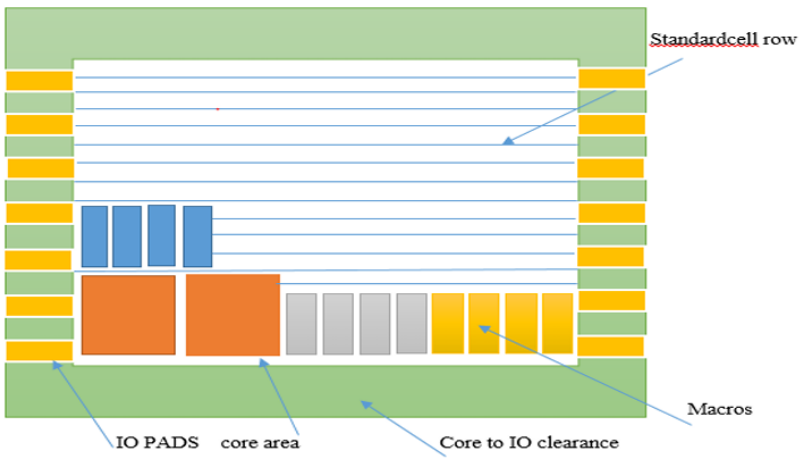

Fig.1.3 Placement of IO pads, Standard cells and Macros hierarchy wise

v. Clock Tree Synthesis (CTS): In CTS a balanced clock network has to be build that distributes the clock signal uniformly across all sinks. The main purpose is to reduce clock skew and latency while keeping the dynamic power. Many different techniques can be used to improve efficiency of whole circuit such as buffer insertion and clock gating.

vi. Routing: We do routing to make sure that all cells should be connected and macros using metal layers, following the design rules like spacing, wire width, and via. Global routing that gives estimated paths for connections, followed by detailed routing to complete wire geometry.

vii. Timing Closure: To get the required performance, including timing-driven and signal-integrity-aware routing are utilized. The process of timing closure ensures that all timing constraints, particularly setup and hold requirements, meet the design and functional scenarios. It is an important step in the physical design workflow. Timing closure analysed the design using Static Timing Analysis (STA) covering many corners (process, voltage, temperature) and modes (functional, test).

1.2 Purpose of the Automated Placement and Routing (PnR) Flow

The implementation of an automated Place and Route (PnR) flows have become essential in physical design because they boost efficiency, precision, and scalability when handling today's increasingly complex integrated circuits.

i. Management of Design Complexity: Contemporary VLSI circuits billions of interconnections that are significantly denser than can be comprehended by humans, rendering hand placement and routing impractical and easy prone to error. To overcome with this complexity, EDA tools used like automated PnR (Placement and Routing), which are based on algorithms that are advanced. These tools ensures that the final design meets all the important constraints such as performance, power, and area.

ii. Enhancement of Time Efficiency: Semiconductor design cycles require fast development and timely delivery. Automated PnR speed up the physical design process by simplifying key steps. It also allows faster iterations and easier improvements in the layout at different stages of design. As a result, the overall turnaround time is reduced, leading to faster product release.

iii. Scalability Across Technology Nodes: Integration density and the overall design as process nodes become tighter. Manual placement and routing is not possible and easy prone to error. Automated PnR flows can easily develop to many design sizes and levels of complexity. Despite the architecture, they support in keeping performance and quality. Maintaining competitiveness and satisfying the evolving needs of the semiconductor industry depend on this scalability.

iv. Reduction of Errors and Verification Assurance: This can easily lead to human errors that can have impact on both yield and functionality. To avoid this, automated flows are designed for detailed verification steps to check design rules and overall design integrity. These checks help to identify violations and early issues in the design cycle itself. As a result, the design becomes more reliable and the expenses for post-layout corrections is minimized.

1.3 Advanced EDA Toolchains for VLSI Physical Design and ASIC Implementation

The physical design phase in VLSI is the step where a synthesized netlist is turned into a layout that can actually be fabricated on silicon. Since this process is quite complex, it depends heavily on advanced Electronic Design Automation (EDA) tools to manage and optimize each stage of the design flow. In the semiconductor field, many tools are used that include Synopsys Fusion Compiler and Cadence Design Systems tools such as Genus and Innovus, which together support a complete ASIC implementation flow.

1.3.1 Cadence Genus

An essential step in the VLSI design workflow, it is a logic synthesis tool that translates RTL designs into gate-level netlists. It utilizes a number of techniques to optimize the design functionality, more particularly by minimizing area, reducing power consumption, and ensuring that timing requirements are met. Its capability to optimize clock and data path equally is one of the features that helps in obtaining better timing closure and increase power efficiency. Moreover, genus has a close relationship with the Cadence Innovus Implementation System, which allows a smoother transition from synthesis to physical design. It also provides a user-friendly interface and strong scripting capability, enabling designers to automate and update their workflows for many semiconductor applications.

1.3.2 Cadence Innovus

It is a physical design tool used for chip manufacturing. It has key steps such as placement, routing, and clock tree to optimize design that creates compact, efficient, and high-performance integrated circuits. It becomes especially important in advanced technology nodes, where issues such as timing closure, signal integrity, and manufacturability are much harder to address. One of its main features is its placement and routing engine, which helps reduce congestion and keep interconnects shorter, so it becomes easier to meet timing and power goals in real designs. Innovus also allows designers to balance performance, power, and area through various optimization options.

1.3.3 Synopsys EDA Suite

An essential step in designing a chip is that it's logic synthesis tool which translates RTL designs into gate-level netlists. These tools are used on larger scale to build complex integrated circuits and complicated System-on-Chip (SoC) solutions in semiconductor industry. RTL code is transformed into an optimised gate-level netlist at the front-end using Design Compiler for logic synthesis. At back-end placement, routing, and timing analysis are synthesized into a single environment. Fusion Compiler offers a unified RTL-to-GDSII flow for physical implementation that converge overall design and power-performance-area (PPA) result.

1.3.4 Synopsys Prime Time

To ensure that timing constraints are met under different operating scenarios, PrimeTime is used for static timing analysis. In addition, IC Validator performs physical verification tasks like as layout versus schematic (LVS) checks and design rule checking (DRC), while StarRC is used for accurate parasitic extraction. These tools are designed to handle the difficulties encountered by existing technological nodes and provides improved scalability, and automation all of which help to better design quality and less time-to-market.

CHAPTER 2

OBJECTIVES

The main objective of this thesis is to study and analyze the physical design phase of VLSI implementation, with a focus on improving performance, power, and area (PPA) metrics. It looks at how modern Electronic Design Automation (EDA) tools help automate important stages such as floorplanning, placement, clock tree synthesis, routing, and signoff. The study also looks at key concepts such as congestion-aware placement, timing-driven routing, and multi-mode multi-corner (MMMC) analysis, which are important for achieving proper design closure across different process, voltage, and temperature (PVT) conditions. In addition, it discusses power optimization techniques using Unified Power Format (UPF) and evaluates integrated tool flows like Cadence Innovus and Synopsys Fusion Compiler for smooth RTL-to-GDSII implementation. Special attention is also given to how UPF helps in managing power intent across different stages of the design flow and supports low-power design techniques.

Along with these aspects, I have also looked into Timing-Driven Non-Default Rule (NDR) strategies to understand how special routing rules affect clock distribution and timing closure. By trying different trunk and leaf configurations, the study aims to show improvements in setup and hold margins. This helps demonstrate that NDR can work as a useful addition to traditional routing and ECO optimization methods.

Through this work, I want to show how tool-driven design convergence, iterative refinement, and advanced timing techniques come together to meet industry-level physical design requirements.

CHAPTER 3

LITERATURE REVIEW

In the paper “A method to speed up VLSI hierarchical physical design in floor-planning” [1] by Yanling Zhou, Yunyao Yan, and Wei Yan address the inefficiencies of conventional iterative design flows by drawing on foundational research, including early automation techniques, nano-scale VLSI design problems, and fixed-outline floor-planning. Previous approaches, such as co-design methodologies for power and ground networks and simulated annealing, enhanced design quality but frequently came with significant computing costs. This work introduces a new approach for building simpler models that substitute non-functional flex fillers for internal logic utilising Active-Logic Reduction Technology (ART). These approaches make timing analysis and placement faster, mainly by shrinking the design's size and complexity without touching the interface logic that has to stay intact. To back this up, we tested it on six large-scale chip designs and saw a clear average gain in runtime along with lower memory usage. The paper provides a workable and scalable strategy that extends the functionality of EDA tools in contemporary VLSI design and adds to the body of current literature by combining ART with hierarchical design.

The article “Optimizing Timing and Physical DRC Performance in VLSI Design: A Comprehensive Approach” [2] by Gundrathi Vinod Kumar, Balwinder Raj, Mandeep Singh, and Aijaz Mehdi Zaidi's work offers a thorough technique for optimising Timing and Physical Design Rule Checks (DRCs) in VLSI design. Building on earlier work in temporal closure techniques, machine learning-based DRC prediction, and placement optimisation, the authors suggest an integrated strategy that uses cutting-edge optimisation algorithms to expedite verification procedures. Unlike earlier approaches that tackled physical design in bits and pieces, this study focuses on improving timing and physical compliance together, at the same time. The findings are backed by extensive simulations and rounds of iterative Engineering Change Orders (ECOs), and the results speak for themselves -DRC violations and verification time both drop noticeably. That's useful groundwork for anyone trying to make designs more efficient and reliable. By bringing physical verification and timing into one unified framework instead of treating them separately, this work pushes the field forward and points toward VLSI design techniques that are both more scalable and more dependable.

According to the study titled “Analysis of CAD tools in VLSI Physical design” [3] by Duttuluru Bharath, A. Jagadeesh, Uday Chaitanya M, and Dr. T. Jaya highlight the tools' functions in handling intricate tasks including floorplanning, placement, routing, and optimisation. The authors look at power, performance, area (PPA), scalability, and usability as their key metrics for judging tools like Synopsys IC Compiler II, Cadence Innovus, and Mentor Graphics Calibre, building on what Chatterjee and Gupta (2024), Singh and Maheshwari (2023), and Liu and Wang (2023) had already found. In addition to highlighting the strengths of each tool-Synopsys in power efficiency, Cadence in routing accuracy, and Mentor Graphics in scalability-the study also covers new developments like the incorporation of AI and machine learning in CAD creation. By providing helpful advice for tool selection based on project-specific requirements and promoting upcoming advancements that improve flexibility, modularity, and cloud-based collaboration in VLSI design automation, this study adds to the body of knowledge.

As presented in “Power Optimization Techniques During Synthesis and Physical Design for a Low-Power RISC-V Design” [4] by Poornima Mohanachandran, Chambamma Koti, and Mahendra Shivaram Gowda provide a thorough analysis of power optimisation strategies used in the physical design and synthesis of a low-power RISC-V CPU. Their work builds on core research in low-power VLSI design, digging into techniques like clock gating, multi-threshold voltage implementation, and dynamic voltage and frequency scaling (DVFS). The authors combine physical design tactics like gate sizing, power-aware placement, and low-power clock tree synthesis with a variety of synthesis-level approaches like constant flop optimisation, gate merging, and multiple supply voltage synthesis. Although technological constraints prevented the use of some more sophisticated approaches, such as power gating and multi-VT, the techniques that were used produced notable gains. Compared to a non-optimized RISC-V design, they found an 11.1% cut in area and a 46.38% cut in power — solid proof that automated EDA tools can genuinely deliver energy-efficient IC designs. That makes this study a useful reference for practical low-power design techniques suited to today's battery-operated systems.

As discussed in “Physical Design: Methodologies and Developments” [5] by Abhay Chopde and Atharva M. Kulkarni offer a thorough literature study that charts the development of hierarchical ASIC design flows and the difficulties presented by diminishing technology nodes in their in-depth investigation of VLSI physical design approaches. They draw attention to how

difficult it is becoming to optimise Power, Performance, and Area (PPA) and how sophisticated EDA tools like Cadence and Synopsys may help meet these demands. In addition to discussing advancements like AI-driven design automation using Cadence's Cerebrus, the study makes reference to fundamental ideas like Moore's Law. It also includes optimisation algorithms like Particle Swarm Optimisation, Genetic Algorithm, and Simulated Annealing, as well as other floorplanning representations like Polish notation, B* Tree, and O-Tree. The authors highlight the significance of strong design flows and flexible approaches in accomplishing effective and scalable VLSI implementations by incorporating knowledge from current research and industry practices.

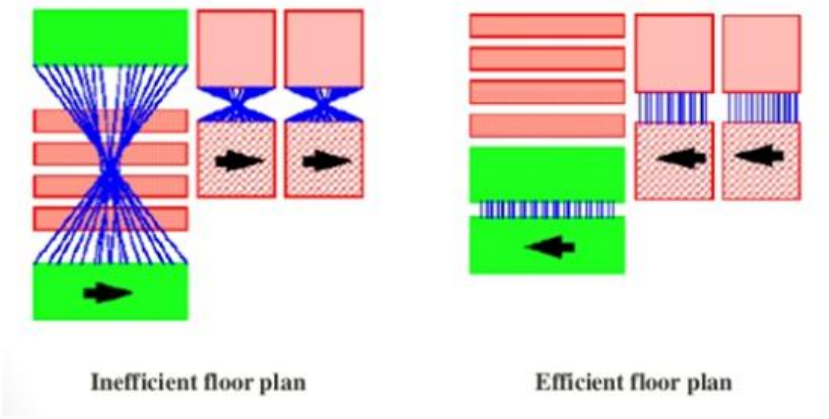

Fig. 3.1 Example of efficient floor plan

The study “Integrated Circuit Conception: A Wire Optimization Technic Reducing Interconnection Delay in Advanced Technology Nodes” [6] by Mohammed Darmi, Lekbir Cherif, Jalal Benallal, Rachid Elgouri, and Nabil Hmina with the goal of lowering connectivity delay in advanced technology nodes, specifically at 7 nm. As a result of increased parasitic resistance and capacitance in scaled nodes, their literature assessment emphasises the increasing dominance of interconnect delay over transistor delay. The authors suggest using the reduced resistance of non-SADP metal layers to reroute essential nets, enhancing timing without affecting area utilisation. This approach builds on earlier research on RC delay sensitivity and routing difficulties. They talk about the drawbacks of conventional buffering and upsizing techniques and make reference to fundamental models like Elmore Delay. The study incorporates knowledge from earlier delay modelling studies. EDA tool capabilities, and metal layer properties, before putting it all into practice using Mentor Graphics' Nitro-SoC tool. Testing across multiple designs shows real improvements in worst negative slack (WNS) and

total negative slack (TNS), which backs up how effective their routing-based optimization strategy actually is.

The research work “Clock Tree Synthesis Techniques for Optimal Power and Timing Convergence in SoC Partitions”[7] by Naveen Kotha, A R Priyarenjini, and Vishnu Priya V examine two cutting-edge approaches: Multi-Source Clock Tree Synthesis (MSCTS) and Multi-Bit Flip-Flop (MBFF) integration, as opposed to depending just on traditional CTS techniques, which frequently have issues with skew and delay in intricate nanometre-scale systems. Drawing on earlier work around clock mesh architectures, timing convergence, and power-aware design approaches, their literature review points out where single-source CTS falls short in modern SoCs. While MBFFs minimise clock sinks and sequential count, which results in lower dynamic power and better area economy, MSCTS combines the robustness of clock mesh with localised subtrees to improve timing resilience and reduce skew. The paper expands upon seminal research in flip-flop architecture and CTS optimisation, providing useful insights into how these methods might improve physical design results in cutting-edge technology nodes.

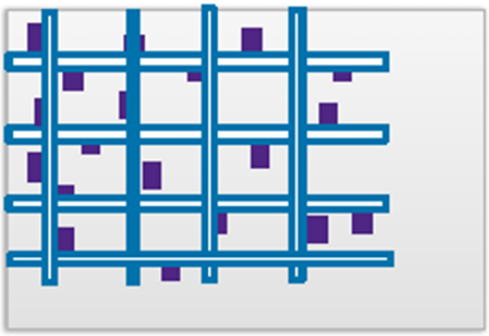

Fig.3.2 Clock mesh

The article “Improving Digital Design PPA (Performance, Power, Area) using iSpatial Physical Restructuring” [8] by Aileen Gumera, Jefferson Hora, Melvin Joey de Guzman, and Susie Maestre's discuss the drawbacks of conventional synthesis and physical design procedures, which frequently have low correlation and necessitate several iterations in order to reach design closure. The study presents the iSpatial flow, a unified engine that combines placement, routing, and optimisation across the stages of synthesis and implementation, building on previous work in physically-aware synthesis, layout estimates, and congestion modelling. The technique improves time predictability and lowers total negative slack (TNS) by utilising physical knowledge early in the design phase and permitting post-placement netlist

reconfiguration. According to experimental results, the iSpatial database handoff is useful in advanced digital IC design because it produces better timing, power, and area results than conventional flows.

A comparative analysis in"RapidPnR: Accelerating the Physical Design for FPGAs via Design-Level Parallelism" [9] by Wanzheng Weng and Pingqiang Zhou start by pointing out that as circuit complexity and logic capacity increase, long runtimes in FPGA physical design become an increasingly difficult problem. By dividing the synthesised netlist into disjoint islands, they offer a novel method that allows for concurrent placement and routing, building on previous work in algorithm-level and design-level parallelism. RapidPnR functions irrespective of front-end design languages and circuit designs, in contrast to previous approaches that were restricted to HLS dataflow architectures. Their approach leans on hypergraph abstraction, grid-based partitioning, and boundary-aware handling of inter-island nets to keep the timing intact. When tested, RapidPnR ran 1.6x–2.5x faster than traditional flows while barely losing any frequency a solid, scalable option for modern FPGA deployment.

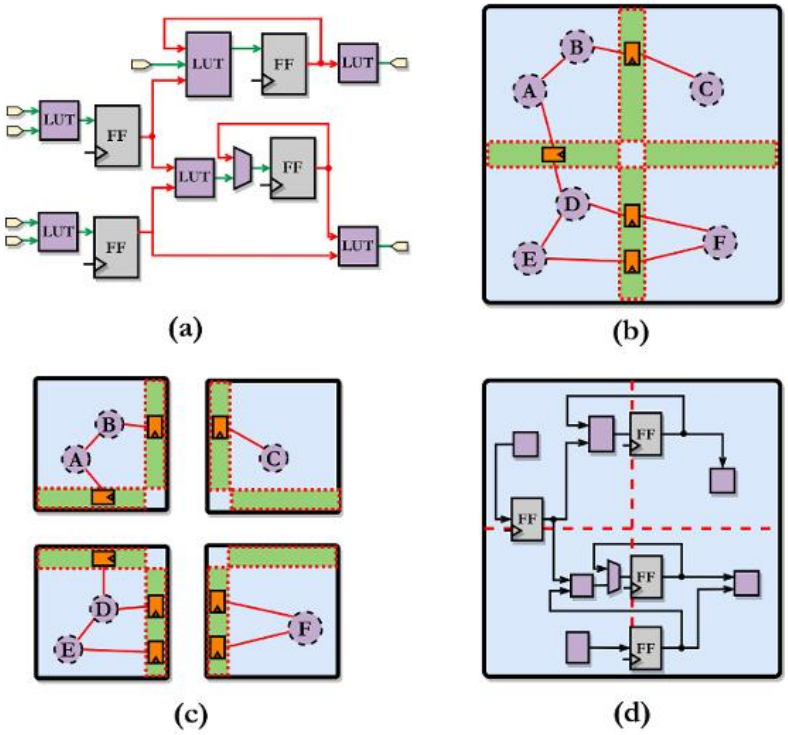

Fig.3.3 An illustration of proposed split-and-parallel physical design flow: (a) the synthesized netlist; (b) the abstract hypergraph and its partition and floorplan results; (c) split and parallel physical design of each island; (d) the complete design after merging islands together.

This research provides an “Analytical Placer for VLSI Standard Cell Placement” [10] by Jianli Chen and Wenxing Zhu ,address the crucial issue of standard cell placement in VLSI physical design in An Analytical Placer for VLSI Standard Cell Placement. This issue directly impacts wirelength, routability, timing, and power consumption. Conventional metaheuristic techniques, including genetic algorithms and simulated annealing, worked well for tiny designs but were unable to scale to big circuits. By recursively splitting circuits into sub-blocks, min-cut partitioning increased efficiency; but, direct wirelength optimisation was constrained by its dependence on cut-size minimisation. Then, analytical methods became the predominant paradigm. While nonlinear programming, utilised in placers like mPL6 and NTUplace3, offered higher accuracy at the expense of runtime scalability, quadratic programming offered quick convex optimisation but only rough wirelength approximations. FastPlace and SimPL are good examples of force-directed placement — they lean on physical analogies to work through iterative refinement and density control. Wirelength comes down nicely with both, but the legalization step afterward is finicky and can't really be rushed. Chen and Zhu took the idea a step further by building a hybrid analytical placer — one that pairs placement-aware best-choice clustering with nonlinear modeling during clustering, then folds in iterative local refinement while declustering. Fast legalization and cell order polishing wrap up the process from there.By bridging the gap between accuracy and scalability, this two-phase framework—multilevel global placement and detailed placement—produces high-quality placements with respectable runtime on benchmark circuits and establishes their work as an attractive alternative to current analytical techniques.

In this paper “Increasing problem of routability in VLSI physical design” [11] by Ying-Yao Huang, Chang-Tzu Lin, Wei-Lun Liang, and Hung-Ming Chen discusses the challenge of resolving short violations during detailed routing, in their paper Learning Based Placement Refinement to Reduce DRC Short Violations. Due to its inability to fully reflect the intricacy of contemporary design guidelines, the traditional dependence on congestion maps from global routing has become inadequate, resulting in a discrepancy between anticipated and actual routability. Previous studies tried to forecast infractions using machine learning frameworks including support vector machines, neural networks, and multivariate adaptive regression splines; others tried density-based placement modifications or probabilistic congestion estimation. Building on these, Huang et al. present a learning-based refinement system that combines placement with a support vector machine model, utilising neighbourhood data, routing demand, and placement attributes to forecast brief violations. Through cell migration

and density modification, short-violation-aware refinement is made possible by feeding the projected violation data back into the placer. Their approach reduces violation-prone nets and balances cell distribution by converting routing information into density factors. In comparison to baseline placers, experimental evaluation on ISPD 2014 benchmarks shows that the approach significantly reduces short violations; in some situations, it even outperforms the ISPD 2014 winning team. By bridging the gap between global and detailed routing and providing a scalable solution to routability-driven placement difficulties, this work demonstrates the efficacy of combining machine learning prediction with placement refinement.

This study examines “Traditional outline-free floorplanning in fixed-outline Floorplanning: Enabling Hierarchical Design” [12] by Saurabh N. Adya and Igor L. Markov addresses the drawbacks of traditional outline-free floorplanning in Fixed-outline Floorplanning: Enabling Hierarchical Design. They then provide a fixed-outline formulation that is more suited to contemporary hierarchical ASIC and SoC design flows. Conventional methods used representations like sequence pairs to optimise area and wirelength, but they were slow and ineffective due to their inability to handle multi-objective trade-offs and absence of incremental move structures. The fixed-outline paradigm exists to mirror real-world hierarchical design constraints, the goal is minimizing wirelength while working within strict outline borders and dealing with soft blocks that don't all have the same aspect ratio.By using slack-based calculations to direct simulated annealing steps, Adya and Markov increase the effectiveness of local searches and make it possible to handle aspect ratios and wirelength minimisation more effectively. By floorplanning designs with tens of thousands of cells in hierarchical contexts, their implementation, Parquet-1, exhibits practical scalability. By connecting traditional theory with contemporary hierarchical design needs, this work demonstrates the increasing applicability of fixed-outline formulations in industry.

This paper presents a comprehensive review of “VLSI design decomposition in Automatic Floorplan Design” [13] by Ralph H.J.M. Otten tackles the macro-placement issue at the highest level of top-down using Schoenberg's theorem to convert module interconnection data into a distance matrix for spatial placement, Otten expands on Heller's previous formulation by using mathematical embedding techniques. In order to ensure balanced module location while taking wiring restrictions and critical nets into consideration, the study suggests using the "Dutch metric" to translate netlist information into proximity measures. This embedding is used to

build slicing structures that produce hierarchical floorplans, which are efficiently represented by the abbreviated tree data structure. The process generates minimum-area floorplans for every chip aspect ratio while taking into account different module flexibility, bonding pad limitations, and wiring space forecasts. In early VLSI CAD research, this study combined mathematical rigour with realistic design considerations, laying the groundwork for autonomous floorplanning.

The present work surveys “PS-FPG: Pattern Selection based Co-design of Floorplan and Power/Ground Network with Wiring Resource Optimisation” [14] by Li Li, Yuchun Ma, Ning Xu, Yu Wang, and Xianlong Hong discuss the crucial problem of IR drop and electromigration in contemporary IC power/ground (P/G) networks in. Conventional methods depended on mesh or tree structures, but they had considerable computing costs during iterative floorplanning or dependability issues. The study presents a pattern selection technique that pre-calculates mesh network patterns in order to get around these restrictions and enable quick signal-integrity assessment without the need for numerous matrix inversions. Furthermore, wire sizing and pin assignment are combined to minimise routing resources under IR drop and EM constraints, and a P/G-aware incremental approach based on B*-tree representation is suggested to intelligently correct violations during floorplanning. The approach dramatically accelerates optimisation while preserving floorplan quality, according to experimental findings on MCNC benchmarks. In order to achieve both power integrity and effective resource utilisation in modern VLSI systems, this work emphasises the significance of co-designing floorplanning and P/G networks.

This study explores “Enhanced Floor Plan Algorithm for Ultra Complex SoCs” [15] by N. Ashok Kumar, K. Nagendra, and P. Shantha Kumar address the increasing difficulties of physical design automation in contemporary SoCs, where leakage power, congestion, and macro placement complexity predominate, in their paper. As instance counts increase, traditional manual macro placement becomes ineffective, resulting in poor core area utilisation and more routing challenges. In order to save manual labour and duration, the study suggests an improved automatic floorplan algorithm that respects design guidelines while maximising power and space. The technique increases metal layer utilisation, decreases congestion, and speeds up the transition to next stages like placement, CTS, and routing by methodically positioning macros and I/O pads. Compared to manual methods, experimental results show improved utilisation, lower congestion percentages, and shorter runtime. In order to accomplish

high-quality IC implementation under extremely complicated design restrictions, this work emphasises the significance of automation in floorplanning.

The present work investigates “Pessimism Reduction in Voltage-Aware DRC using Simulation Results for Functionally Correlated Nets” [16] by Sonipriya Singh, Pardeep Juneja, Rahul Tewari, Sachin Shrivastava, and Sankalp Srivastava discuss reliability issues in advanced VLSI technologies where decreased wire spacing increases susceptibility to dielectric breakdown in. Conventional voltage-aware design rule checks (DRC) are built around static voltage variations, and that tends to produce spacing requirements that are overly pessimistic — wasting valuable chip surface in the process. The research digs into cases of coupled nets, like synchronous or topology-driven signals, where these static checks end up inflating voltage discrepancies compared to what actual simulations show. In order to get around this, the authors suggest a methodology that uses markers and limits to transmit realistic voltage changes throughout schematic, layout, and sign-off stages while integrating electrical simulation data into physical design. Testing on analogue systems shows a real drop in voltage-aware DRC violations, which speeds up the design cycle without sacrificing reliability. What this paper really shows is that simulation-driven methods can cut down on unnecessary pessimism in voltage-aware spacing rules, making better use of advanced process nodes in the process.

In this article “A Power Delivery Network (PDN) Engineering Change Order (ECO) Approach for Repairing IR-Drop Failures after the Routing Stage” [17] by Tsu-Wei Tseng, Chang-Tzu Lin, Chia-Hsin Lee, Yung-Fa Chou, and Ding-Ming Kwai discuss the difficulty of mitigating IR-drop problems in integrated circuits at the signoff stage, where routing resources are limited, in their paper. In crowded layouts, conventional PDN repair techniques like wire widening or stripe addition frequently fail, necessitating expensive redesign revisions. In order to get around this, the study suggests an eco PDN technique that uses a greedy-Pareto-optimal (GPO) selection strategy to find effective locations for reinforcement and introduces detour-shaped ECO power rails. This method avoids the need to go back to previous design stages by striking a compromise between IR-drop improvement and minimal routing resource use. With respectable runtime and resource consumption, experimental findings on a large image signal processor design show a considerable reduction in IR-drop, improving the worst-case drop from 9.34% to 3.84%. With possible extensions to time and temperature issues in future work, the study emphasises eco PDN as a workable method for late-stage PDN repair.

This paper presents a comprehensive review “VLSI design in A Novel Clock Distribution Technology – Multisource Clock Tree System (MCTS)” [18] by Nikhil Patel examines clock distribution techniques essential to high-performance. Despite being easy to build and power-efficient, conventional clock trees have low tolerance to on-chip variation (OCV), clock skew, and jitter. Conversely, clock meshes nearly remove skew and increase yield because of OCV immunity, but because of their dense grid structure, they utilise a lot more power. MCTS is presented as a hybrid method that combines the performance and OCV tolerance of clock meshes with the low power and simplicity of traditional trees. It relies on a coarse mesh fabric, a pre-mesh H-tree, and moderately sized clock trees to strike a balance between skew, power, and design complexity. This hybrid setup holds up well across multiple corners, keeps skew low, and gives more flexibility for complex floorplans — making it a solid option for today's high-frequency designs.

According to the study titled “Crosstalk Noise Optimisation by Post-Layout Transistor” [19] by Masanori Hashimoto, Masao Takahashi, and Hidetoshi Onodera present a transistor size framework in size to reduce capacitive coupling noise following careful routing. This solution ensures that no new noise issues arise by downsizing aggressor drivers while maintaining interconnect structures, in contrast to conventional routing or buffer insertion techniques. To prevent pessimistic superposition of noise, the technique uses time window analysis and a 2-π noise model with enhanced aggressor characterisation. In order to balance delay limitations with efficient noise reduction, an optimisation algorithm ranks aggressors according to timing slack and noise contribution. The usefulness of post-layout transistor scaling is confirmed by experimental results on benchmark circuits that show a maximum noise voltage reduction of more than 50% without delay penalties. A strong late-stage optimisation method that improves reliability in deep submicron VLSI designs is highlighted in this paper.

CHAPTER 4

SYNTHESIS

The synthesis phase in the VLSI design flow converts high-level RTL descriptions into gate-level netlists that are suitable for physical implementation. It requires files as input that collectively define the design intent, constraints, technology details, and verification settings. Each of them has its own role in the synthesis tool to produce a correct and well-optimized netlist in terms of functionality and performance.

During the synthesis phase, the tool does the logic optimization, technology mapping, and timing estimation. The main goal is to ensure the final netlist functions correctly while also improving performance, power, and area (PPA). This netlist is then used as the base for later physical design stages such as placement, routing, and timing closure. The tool also provides timing, area, and power reports, which offer an early understanding of the design and enable improvements where needed.

In simple terms, synthesis is a combination of logic optimization and mapping. In Synopsys, the translation happens when the design or input files are read, while logic optimization and mapping are handled during the compile stage.

4.1 Synthesis Flow

i. Develop HDL Files: It starts with writing HDL code, either in Verilog or VHDL, which explains how the digital circuit should behave at either a behavioral or structural level. This code then serves as the base for synthesis, and provides a clear picture to tool how exactly the circuit needs to be operated.

ii. Specify Libraries: From there, you set up the technology libraries the tool will need the link library for referencing existing cells, the target library holding the standard cells used during optimization, and the symbol library for schematic and visualization purposes. Getting these libraries defined properly is what lets the synthesis tool know exactly which components it has available to map the HDL design into actual physical gates.

iii. Read Design: The HDL design is input to the synthesis tool which opens the source code and prepares it for analysis. The program creates an internal representation of the design

which will be refined later after parsing through the HDL files and marking any syntax errors.

iv. Define Design Environment: The next step is to configure the design environment to match real operating conditions. This includes configuring voltage and temperature ranges, creating wire load models to estimate interconnect delays and defining input/output delays to capture the way the circuit interacts with the outside world. These parameters drive the synthesis tool to create a design that really maintains under practical boundaries.

v. Set Design Constraints: From there, design constraints come into play to keep both the physical and performance sides of the circuit in check. Rule-based constraints — such as maximum transition time, capacitance, and fanout — help maintain electrical reliability. In contrast, optimization constraints such as area limits and timing requirements push the tool to strike the right balance among performance, power, and silicon usage.

vi. Select Compile Strategy: At this stage, the compilation strategy is selected. Designers can use a top-down approach for compiling the whole design at once, or a bottom-up approach where individual modules are compiled and integrated. The selection depends on the complexity of the design, and the optimisation goals, as these factors influence the quality of the final results.

vii. Optimize the Design: The synthesis tool then runs optimisation using commands like compile or compile ultra. This stage takes the HDL description and maps the design onto standard cells, resulting in a gate-level netlist that minimises the area and power consumption and meets the timing requirements.

viii. Analyze and Resolve Problems: Reports are generated after optimisation to assess the designs. Timing reports check for violations such as setup or hold failures, area reports check silicon usage, and power reports estimate consumption. Any problems discovered are fixed by changing constraints or improving the design so that it meets specs.

ix. Save the Design Database: Finally the optimised design is saved into a database by commands such as write. This database is ready for downstream processes such as placement, routing and signoff verification, and contains the gate-level netlist, constraints and environment settings.

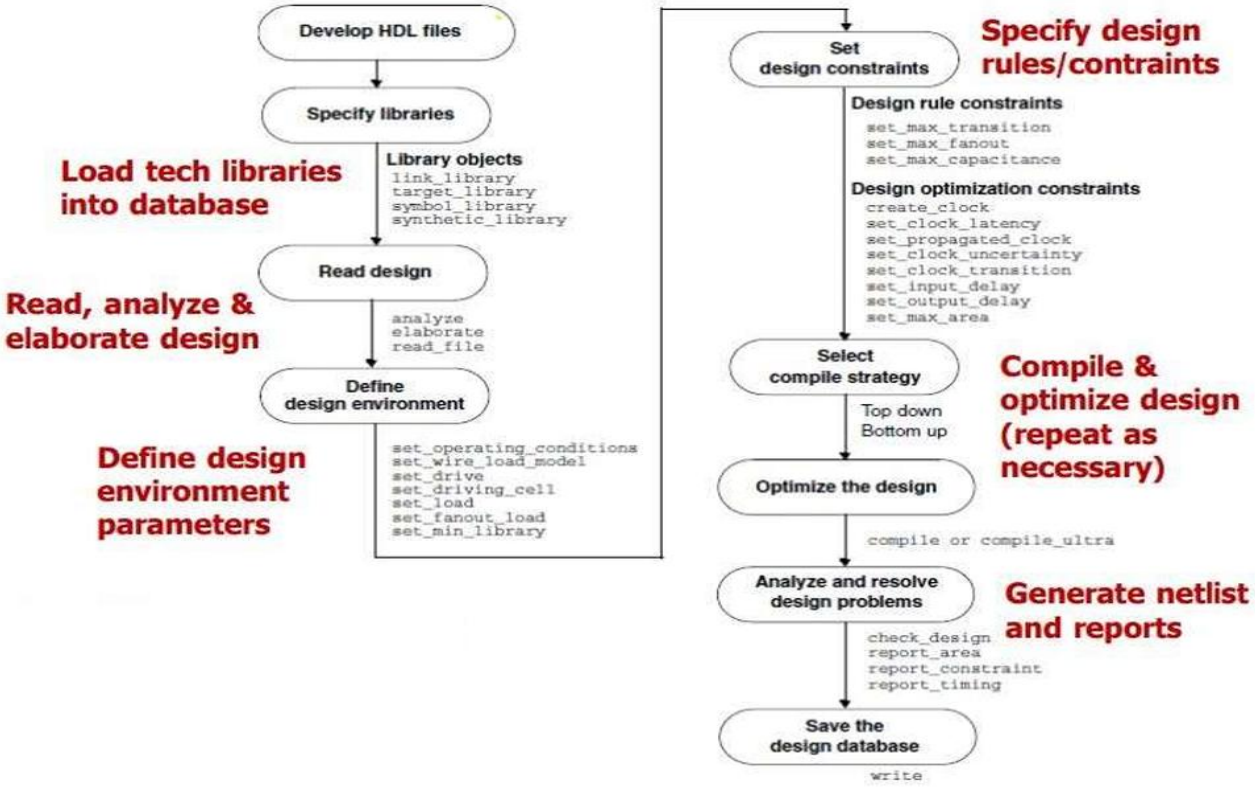

Fig.4.1 Synthesis Flow

4.2 Input files required for Synthesis

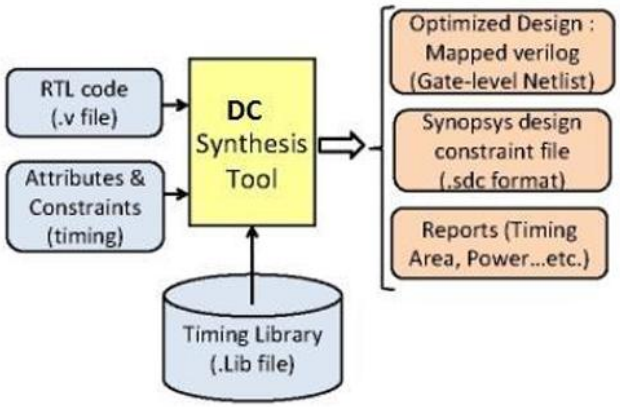

Fig.4.2 Input files for synthesis

i. Verilog file (.v): The Verilog file contains the RTL source code that describes the functionality of the digital system. It has modules, signal declarations and control logic written in Verilog HDL. This file serves as the primary input to the synthesis process, which translates the design from a behavioural description to a gate level representation.

ii. Logical Libraries (.lib): Logical libraries are usually provided in the Liberty format (.lib) and include timing, power, and area details for standard cells and macros. These libraries help the synthesis tool map RTL code to technology-specific gates while optimizing key factors like power, performance, and area (PPA). An ASIC designer typically gets the .lib file from a standard cell library team. The information inside a .lib file is usually divided into two main parts. The first one contains data common to all standard cells.

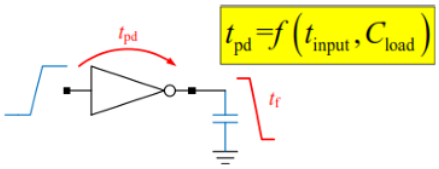

Fig.4.3 Propagation delay of a Inverter

The common part of Lib file contains

- Library name and technology name.
- Units (of time, power, voltage, current, resistance and capacitance).
- Value of operating condition (process, voltage and temperature) – Max, Min and Typical

In the second part of Lib file, it contains cell-specific information for each cell. The part of Lib file which contains cell-specific information is shown below:

Cell-specific information in Lib file is mainly

- Cell name
- PG Pin name
- Area of cell
- Leakage power in respect of input pins logic state

Timing and power parameter of a cell is obtained by simulating the cell in a variety of operating conditions and data are represented in the Lib file.

There are two main techniques to characterize a cell and generated the Lib file.

- CCS (Composite Current Source)
- NLDM (Non-Linear Delay Model)

In CCS technique current source is used whereas in NLDM technique voltage source is used to model and derive the Lib parameters. Based on CCS and NLDM technique used for characterizing the cell, we call the corresponding Lib file is CCS Lib file and NLDM Lib file. Since CCS technique is having more controlling parameters as compare to NLDM technique, So CCS Lib file is more accurate. NLDM Lib file has lesser run time mean fast run compare to CCS Lib and also the size of the NLDM file is lesser than that of CCS Lib file.

iii. Physical Libraries (.lef): Physical library files like Library Exchange Format (.lef) files and frame view definitions describe the physical information of standard cells and macros. They basically include details such as cell size, pin locations, and routing blockages. These files are mainly used during floorplanning and placement in physical design, but they can also be used during synthesis to get an early idea of congestion and overall physical layout awareness.

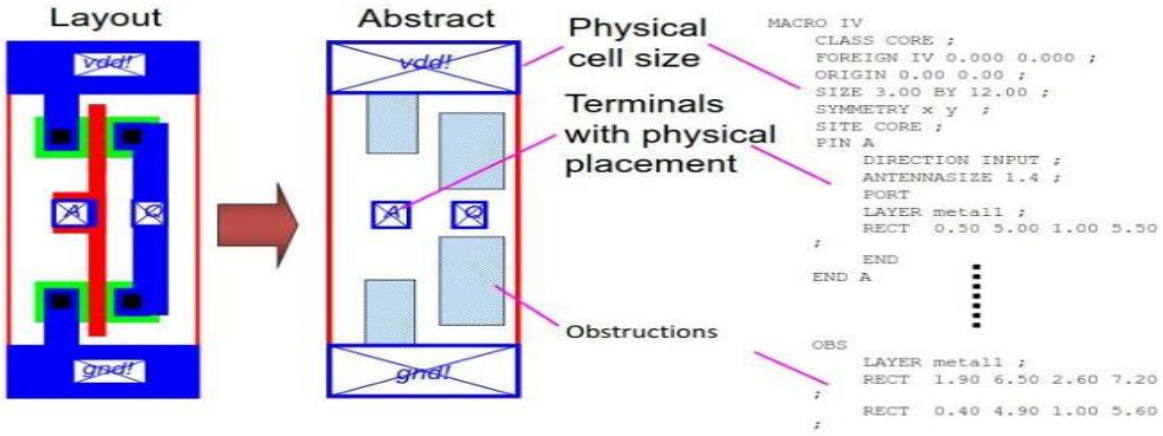

Fig.4.4 Physical libraries file (.lef)

iv. Technology File (.tf): The technology file defines process-specific parameters such as routing rules, layer definitions, via characteristics, and design rules. It provides the synthesis tool with information about the manufacturing process, enabling accurate estimation of wire delays and layout feasibility. A sample snapshot is given below to show the information under technology LEF part.

Technology LEF part contains the following information

- LEF Version (like 5.7 or 5.8)
- Units (for database, time, resistance, capacitance)

- Manufacturing grids
- Design rules and other details of BEOL (Back End of Layers)
- Layer name (like poly, contact, vial, metal1 etc) 15
- Layer type (like routing, masterslice, cut etc)
- Preferred direction (like horizontal or vertical)
- Pitch
- Minimum width
- Spacing

v. TLU+ File (.tlup): The TLU+ (Timing Library Unit Plus) file contains pre-characterized parasitic data used for accurate wire delay modelling. It goes beyond simple RC (Resistor-Capacitor) models and accounts for more complex effects. The core of the TLU+ format is a set of multi-dimensional tables. These tables store pre-calculated values for capacitance and resistance per unit length of the wires. The "Plus" in TLU+ indicates that it often includes more advanced modelling parameters beyond basic RC. It is specific to the manufacturing process and the different metal layers available in that process. Each metal layer will have its own set of characteristics defined within the file. Tools for parasitic extraction (like StarRC, Calibre xACT) and static timing analysis (like PrimeTime, Tempus) consume the TLU+ file to accurately determine the delay introduced by the interconnect and to analyze signal integrity issues like crosstalk. It supports early timing analysis by providing resistance and capacitance values for various metal layers and routing scenarios. This file is particularly important for advanced nodes where interconnect delay dominates overall timing.

   Generally, it contains:

   a) Metal Layers: Definitions of each metal layer, including its thickness, width, and spacing rules.
   b) Dielectric Layers: Properties of the insulating layers between the metal layers, such as their dielectric constant.
   c) Capacitance: Tables that provide capacitance values based on factors like:
      - Width of the wire: Wider wires generally have higher capacitance.
      - Spacing to neighbouring wires: Closer wires have higher coupling capacitance.
      - Overlap with wires on adjacent layers: Inter-layer capacitance.

- Proximity to the substrate: Ground capacitance.

d) Resistance: Tables that provide resistance values based on the width and length of the wire (resistance is inversely proportional to width and directly proportional to length).

e) Advanced Modelling Parameters: Depending on the "Plus" part, it might include parameters to model effects like:
   - Skin effect: At high frequencies, current tends to flow on the surface of the conductor, increasing resistance.
   - Proximity effect: The effect of nearby conductors on the current distribution in a conductor.

4.2.1 Import Design

a) Takes logical libraries, physical libraries and technology file as input. This contains information related to all the standard cells and macro references that are to be used in the design.

b) Takes RTL file as input and performs elaboration of the design.

c) Allows to put restriction on which library cells are not to be used in the design. This is also known as “don’t use” list, which can be specified at this step.

d) Also, since mostly the designs are divided into small partitions and handled in parallel, information related to interface paths with other partitions or feedthroughs are also consumed in this stage. This helps in getting a better picture of the partition at full – chip level.

4.2.2 Unified Power Format File (.upf)

The Unified Power Format (UPF) file is what carries a circuit's power intent in low-power VLSI design — it lays out the power domains, voltage sources, level shifters, isolation cells, and retention cells. EDA tools like Cadence Genus and Synopsys Design Compiler pick up on these UPF files and automatically slot in the necessary low-power cells during synthesis. The result is a design that actually meets the power management requirements it's supposed to and runs correctly across all its different power domains.

The need for UPF arose from the increasing complexity of power management techniques employed in modern SoCs (System-on-Chips). These techniques include:

- Multiple Power Domains (MPDs): Dividing the chip into regions that can operate at different voltage levels or be completely powered down to save energy.

- Voltage Scaling (AVS/DVS): Dynamically adjusting the supply voltage of a power domain based on the operating frequency and workload.
- Power Gating: Completely turn off the power supply to inactive blocks of the circuit to minimize leakage current.
- Clock Gating: When the clock signal is restricted to idle functional units to reduce dynamic power consumption.
- Power State Retention: Preserving the state of a block (e.g., register values) when it is powered down, allowing for a faster return to operation.

It would be difficult for designers to explain the power intent to different tools and ensure correct execution without the use of a standardized way to explain these power elements. By introducing detailed terms of defining power domains, their voltage levels, power states, power switches, level shifters, isolation cells, and behavior of the circuit during power transitions, UPF takes on this issue.

A UPF file is structured around several key concepts and commands that collectively define the power architecture and behavior of the design:

- Power Domains: Power domains are the basic building blocks of power management. A power domain is a part of a design that shares a common power supply. With UPF, designers can define multiple power domains, each having its own name, set of instances (modules or cells), and operating voltage. The create_power_domain command is used for defining these domains in the design.
- Power States: Within each power domain, the design can operate in different power states such as “ON,” “OFF,” “RETENTION,” or other user-defined states. Each state corresponds to a specific power consumption level and may also use a different operating voltage. These states are defined within a power domain using the create_power_state command.
- Power Switches: These are usually transistors or dedicated power-management transistors designed to control power distribution across different power domains. In UPF, designers define power switches and describe their type (header vs. footer switches), their position in the hierarchy, and the power domains they control. This is set up by the create_power_switch command.

- Power Nets and Ground Nets: UPF allows the express declaration of power and ground networks in the design. This is important for accurately modeling power distribution and properly connecting power management circuitry.
- Voltage Sources: UPF allows specifying voltage sources and their nominal voltage levels. These voltage sources are then associated with power domains. The create_voltage_source command is used for this purpose.
- Level Shifters: Level shifters are required when signals cross the boundaries between power domains operating at different voltage levels, to ensure proper signal integrity.
- Isolation Cells: Isolation cells are inserted at the boundaries to prevent unwanted current leakage or logic corruption when a power domain is turned off.
- Retention Registers: Retention registers are used for power states where a block power has to be stored. UPF allows designers to identify registers that need to retain their values during power down and specify the power domain responsible for their retention power supply.

4.2.3 Initialize Floorplan

- Technology file details are read during this stage such as routing layer direction of all metal layers.
- Creates site array definitions.
- Defines design boundaries using def file.
- Core offset is defined (if any).
- Defines voltage area.
- Route track pattern for specific layer can be defined. It includes the routing direction, offset, track type, color if applicable, width, spacing, and number of repeated pattern.
- Pre-synthesis floorplanning details are taken as input. This includes placement constraints, NDR specifications, blockages, powergrid details, macro placement, I/O placement, voltage area, etc.
- Logical net connections are created for powergrid.

4.2.4 Read timing constraints

- Modes, corners and scenarios are specified using MCMM file.

- TLU+ file is consumed at this stage. Depending on PVT settings for each corner, parasitic extraction is done.
- SDC file and OCV data are read.
- The SAIF file is read for dynamic power scenarios.
- Path groups such as such as input, output, feedthroughs and custom path groups are created.

4.2.4.1 Multi-Corner Multi-Mode (MCMM)

With a growing variety of modes and corners in sophisticated technology, the variety of nodes has become a major difficulty for designers. Because designs must be simultaneously optimized for an increasing number of modes and corners, IC physical design tool issues are at the leading-edge nodes. For most tools, a distinct abstraction model is needed for each mode/corner scenario to achieve top-level closure across timing, signal integrity, power, and manufacturing design requirements. Running fewer instances with larger timing margins is a popular approach. This strategy results in delayed time-to-market, larger die sizes, and higher power consumption, all of which reduce product competitiveness.
For instance, if the analysis tool is unable to handle all modes, corners, power states, and complex clocking simultaneously, issues may occur when attempting to close the chip's timing. The block-level implementations are frequently examined only for "best case/worst case" functional timing to save time. However, some modes and corner combinations might not meet constraints at sign-off. Before the silicon can be taped out, engineers must correct these cross-corner or cross-mode failures through laborious ECO loops.

Modes: A mode defines a specific operating state of the chip. This can include:

- Functional Modes: Normal operation, high-performance mode, low-power mode, idle mode, etc.
- Test Modes: Scan testing, Built-In Self-Test (BIST) modes, JTAG boundary scan. Compared to functional modes, these modes have different clocking ways and limits.

Corners: A corner is a particular set of functional or manufacturing conditions for a chip.
The following are the primary components of corners:

- Process (P): Transistors can operate faster or slower than usual because of variations in the manufacturing procedure. Fast-fast(FF), Slow-slow(SS), and typical-typical(TT) are common process corners, as are combinations like fast-slow(FS) and slow-fast (SF) to account for inter-die changes.
- Voltage (V): The supply voltage may deviate from its nominal value due to the characteristics of the power delivery network and external influences. High-voltage and low-voltage corners are accounted for.
- Temperature (T): The operating temperature of the chip can range from cold to hot.
- Interconnect Corners: Variations in the manufacturing of metal layers can affect the resistance and capacitance (RC) of the interconnects. Corners like maximum capacitance (MaxC), minimum capacitance (MinC), maximum resistance capacitance (MaxRC), and minimum resistance-capacitance (MinRC) are used for parasitic extraction.

Scenarios: An MCMM configuration defines multiple scenarios, where each scenario is a unique combination of one or more modes and one or more corners. For example, you might have a scenario for "Functional Mode at High Voltage and Slow Process Corner" or "Scan Test Mode at Nominal Voltage and Typical Process Corner."

vi. Constraints File (.sdc): Timing and design constraints are described in the Synopsys Design Constraints (SDC) file and are critical for the successful execution of the synthesis procedure. Specifications in the SDC file include clock definitions, input/output delays, false paths, multicycle paths, and timing exceptions.

Inside the sdc file, some important constraints are as follows:

a) Set driving cells: It defines how strongly input or inout ports are driven by cells from the technology library. These commands link a library pin with input ports so that delay can be modeled more accurately during timing analysis.
b) Set load: This command assigns load values to specific ports and nets in the design, with the load measured using whatever capacitance unit the file defines, so the values stay consistent across the whole design.
c) Set maximum fanout: The maximum fanout load is defined for a specific input port or for the design as a whole.

d) Set maximum transition: This command sets the maximum transition time, which is treated as a design rule. It can be applied to a clock port, a specific input port, or the entire design to control signal slew limits.
e) Create clock: The create_clock command is used to define a clock in the current design by setting the specified source objects as the clock source.
f) Create generated clock: The create_generated_clock command is used to define a generated clock object. It can be assigned to a specific pin or port, and it is derived from a master clock, meaning any change in the master clock will automatically be reflected in the generated clock as well.
g) Group path: Groups are basically a collection of paths or endpoints that get used for cost function calculations during optimization. They let you target specific sets of paths for optimization, even when other groups elsewhere in the design have bigger violations. And once endpoints are defined, every path leading to them gets pulled into that same group for analysis and optimization.
h) Clock uncertainty: Once the clock defined, the next step is adding clock uncertainty to account for variations in the clock network. This basically gives you a safety margin for things like clock source inaccuracies and the delays that come from the clock distribution network not behaving ideally. The command can define either inter-clock uncertainty or general clock uncertainty, and either way, it helps keep the timing analysis reliable.
i) Clock latency: Clock latency refers to the delay a clock signal experiences as it travels from the clock source to the clock pin of a sequential element. There are two types of clock latency: network latency and source latency. Network latency is the default case and represents the time taken for the clock to move from the clock definition point to the register clock pin. Source latency, which is set using the -source option, accounts for the delay from the actual clock origin up to the clock definition point in the design.

4.3 Output files generated after Synthesis:

The synthesis process transforms RTL code into a gate-level representation optimized for timing, power, and area. Upon completion, the synthesis tool generates the following output files:

i. Gate-Level Netlist (.v): This Verilog file contains the structural description of the design using standard cells from the target technology library. It serves as the primary input for physical implementation.

ii. Constraint File (.sdc): The updated Synopsys Design Constraints file reflects refined timing constraints post-synthesis, including clock definitions, input/output delays, and timing exceptions.

iii. Reports (.rpt): Various synthesis reports are generated, such as timing analysis, area utilization, power estimation, and design warnings. These help evaluate the quality of results (QoR) and guide optimization.

iv. Design Database (.db): A binary database format used by Synopsys tools to store the synthesized design, enabling seamless transition to physical design tools like IC Compiler or Fusion Compiler.

v. Equivalence Checking Files (.svf): These are the files which will be used in formal verification just to ensure that there is no mismatch between the RTL and the netlist.

vi. Log Files (.log): Execution logs capturing tool commands, messages, and synthesis flow details for debugging and documentation.

These outputs collectively enable the transition from logical synthesis to physical implementation, ensuring that the design is functionally correct and meets specified constraints.

CHAPTER 5

OPTIMIZATION

5.1 Static Timing Analysis

Static timing analysis (STA) refers to a type of simulation that estimates the timing of a digital circuit without necessitating the simulation of the entire circuit. STA ensures that the timing paths within the system meet timing constraints across various operating frequencies and voltages. STA analyzes all paths from the start point to the endpoint that are present in the design. It then compares the paths against the constraints defined for each path. All paths are constrained by the clock period and the timing characteristics of the circuit's primary inputs and outputs. The timing paths can be divided into data paths, clock paths, clock-gating paths, and asynchronous paths. For a data path, the start point can be either the input port or the clock pin of the flip-flop/latch, and the end point can be the data input pin of the flip-flop/latch or the output port of the design. The types of paths are shown in Table 1 and figure 5.1 shows the start- endpoint of different types of paths.

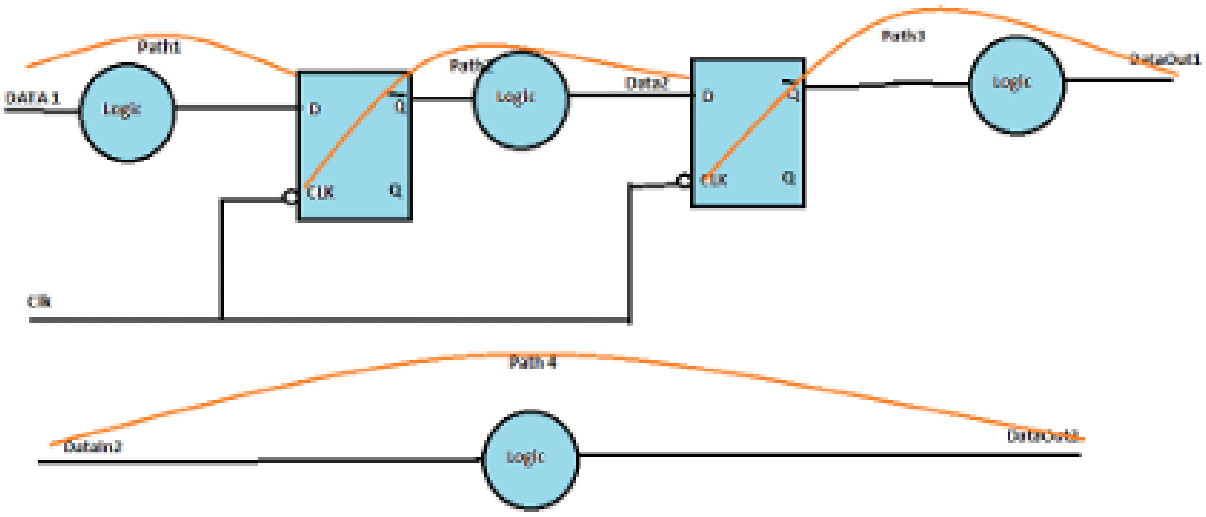

Fig.5.1: Types of Logical path

Table 1: Path Types and Description

PATH	TYPE	DESCRIPTION
Path1	Input pin/port to flip-flop	Starts at an i/p port and ends at data i/p of sequential element
Path2	Input pin/port to Output pin/port	Starts at the clock pin of sequential element and ends at the data input of sequential element
Path3	Flip-flop to Flip-flop	Starts at the clock pin of a sequential element and ends at an o/p port
Path4	Flip-flop to Output pin/port	Starts at an i/p port and ends at an o/p port

5.2 Setup Time

Setup time is the minimum amount of time before the active clock edge during which the data must remain stable so that it can be captured correctly. If a design meets the setup time requirement, it means the data launched on the previous clock edge is reliably captured at the current edge without errors.

5.3 Setup Violation

Consider there is a setup timing violation for the data path (Sequential to Sequential) shown in figure 6.3. Here, clock arrives simultaneously at both sampling and generating FFs. A setup violation happens when the data generated at a clock rising edge fails to arrive at the sampling FF within one clock cycle. i.e., the path fails to satisfy the following expression.

The setup violation can be fixed by:

a. Clock Tuning: Delaying the clock (pushing) to sampling FF or fastening (pulling) the clock to generating FF can relax the timing constraint by providing more time (Tclk) for the data path to complete.
b. Reducing the data path delay: Data path can be made faster by increasing the driving strengths for the standard cells in the data path. Upsizing will decrease the cell delays and also improve the output signal slopes.

i. The simplest way is to replace the SVT/HVT cells by LVT/ULVT cells, reducing delay and improving drive strength.This ECO-stage change avoids rerunning PnR but increases leakage power.

ii. Upsizing the cell to improve driving strength and cell delay but both area and power is compromised.

iii. By adding buffers, the driving capacity of cells is extended since each can only handle limited output capacitance. However, this increases area usage and power consumption.

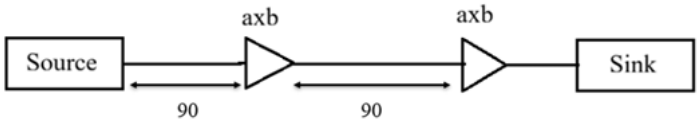

Fig 5.2: Buffer Insertion with Equal Driving Strength for Distance Partitioning

iv. Routing rules include avoiding usage of via, reducing unnecessary detours in nets, and beneficial for upper metal layers, which offer lower resistance.

v. Choose best placement to improve net delay.

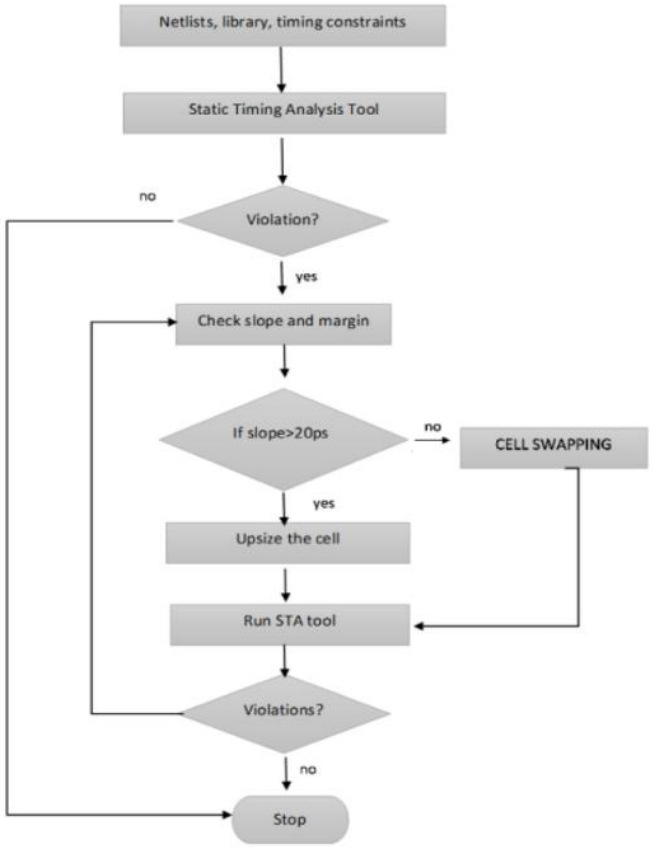

Fig.5.3: Flowchart for Setup Fix

A flow chart for fixing the setup violations is presented in figure 5.3 Besides using these methods for fixing setup violations we are using two more methods. Details of them are given below:

1. Bounded Logic: For a particular input or output port, there is a large amount of logic associated with it. Sometimes, the number of cells in the logic is so large that they are spread across our design. Moreover, if the cells on the critical path are, in turn, driven by logic placed far from the port, the signal has to travel a longer distance. This results in a larger RC delay in the path. Moreover, the cell's delay increases as the load at its output increases.

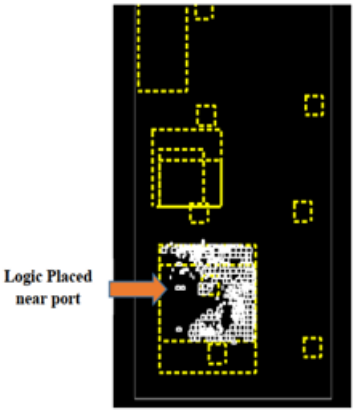

Fig.5.4: Bounded Logic

2. Unbounded Logic: As seen in the below figure, the logic is getting placed far away from the port. Thus, it will result in interconnects getting routed in longer lengths. This in turns means more R and more C leading to poor path delay. Now if we bound the logic associated with

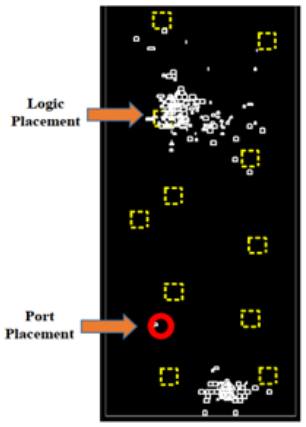

Fig.5.5: Unbounded Logic

this particular port to someplace around the port, it will result in better placement, less length of interconnects and thus lesser delays. Situation after bounding the logic near port is shown in figure 5.4.

This helps in improving our timing to a great extent and helps us in meeting setup requirements.

5.4 Hold Time

Hold time is the minimum amount of time after the active clock edge during which the data must stay stable. In simple terms, each sequential element needs the data to remain unchanged for a short period after the clock edge so that it can correctly capture and store the value.

The hold violation can be fixed by:

a. Clock path violations: Push launch clock (by adding buffers) to increase the clock latency of launch Flop/ Pull the capture clock (by removing buffers) to decrease the clock latency of capture flop.
b. Delaying the data path: Data path can be delayed by adding buffers to the path or by replacing normal voltage cells with high- Vth cells. Such cells have larger delay due to reduced voltage swing.
   i. Use low Vt cells: Swapping LVT/ULVT cells with SVT/HVT cells helps increase data path delay, but the setup margin must be carefully preserved.
   ii. To address larger hold violations, delay cells with higher intrinsic delay can be used instead of inserting multiple buffers. While effective, this approach comes at the cost of increased area and power.
   iii. By inserting low drive-strength buffers, which introduce higher delay, the data path delay can be increased.
   iv. Hold violations can be mitigated by increasing wire length, introducing routing detours, or using lower metal layers.
   v. Split the driver which has high fan-out.

5.4.1 Fixing Max Transition Violations

Maximum transition is simply the longest time a signal is allowed to rise or fall in a design. It is written in the technology library to keep signals switching fast enough, avoiding extra delay or weak performance.

A max transition violation occurs when a signal's slope increases from the defined threshold. This usually happens due to heavy capacitive loading, long interconnects, or weak driving strength of the source cell. These need to be fixed:

i. If the violation is small, replace the driver's HVT cell with an LVT cell to boost current and improve drivability.
ii. Change the driver cell size to boost current, since it scales with the W/L ratio, thereby increasing drive strength as needed.
iii. Clone the driver to split fanout, distributing the load across multiple drivers for better performance.

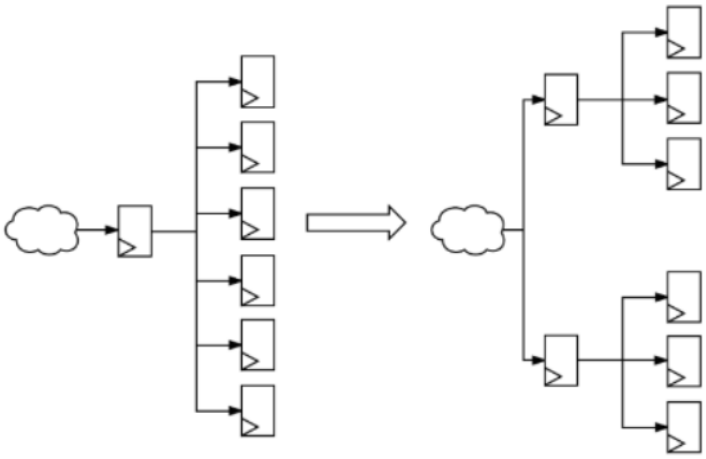

Fig. 5.6: Splitting the fanout

5.4.2 Fixing Max Capacitance Violations

Maximum Capacitance represents the maximum value of the load that the driver can take on its output pin as defined in the technology library. This will make sure that the driver switches within the time boundaries without creating any delay.

Fixes:

1. Increase driver cell size.
2. Clone the driver to split fanout.
3. Use buffers to divide fanout.

5.4.3 Fixing Crosstalk Violations

Crosstalk in VLSI circuits is the interference caused due to parasitic capacitance and inductances between the nearby interconnects. If there are two nets that are adjacent to each other, then the switching activity in the aggressor net will cause some sort of disturbance in the

victim net. With advancements in technology nodes where wire spacing is narrow and the interconnect density is very high, crosstalk plays a significant role.

Fixes:

1. Reduction of coupling capacitance can be achieved through track assignment and removal of the aggressor from the vicinity of the victim.
2. Victim nets need to be split into smaller lengths is another effective solution.
3. By increasing or decreasing the size of the victim and aggressor drivers will provide the required balance.
4. Using the non-default rules (NDRs)- for example, the use of wider metals (2w) that decrease resistance and latency and larger spacing/pitches (2s, 2p) that helps to reduce crosstalk.
5. Ground shielding of victim nets to eliminate coupling noise.
6. Placement of buffers in victim nets should be proper.

CHAPTER 6

Timing-Driven NDR Analysis

6.1 Introduction to NDR

While working on my project, I examined the importance of the Non-Default Rules (NDR) in the VLSI physical design process. By default, routing rules specify wire width, spacing, and pitch. But certain nets, such as clock or high-speed data lines, need enhanced routing parameters. For the solution of the problem, I have used NDR settings and studied their effects on timing closure.

6.2 Effects of NDR

As observed from my experiments, some of the measurable effects of using NDR include:

- Narrower wires resulted in reduced resistance, hence lower propagation delay.
- Bigger spacing between the wires minimizes the effects of crosstalk, hence ensuring stable timing.
- Combining both width and spacing gave the optimal solution between delay reduction and noise tolerance.
- The negative impact of this process is that, it requires more routing area and congestion.

6.3 Purpose of NDR

The purpose of introducing NDR in my design flow was:

- Evaluate timing within critical paths.
- Mitigate noise and crosstalk on sensitive nets.
- Balance the clock distribution network for minimal skew.
- Enhance overall design stability at advanced technology nodes.

6.4 Rules of NDR

Examples of the NDR methodology includes:

- Make wires wider (2w): If you double the width of your traces, you lower their resistance. This helps electricity flow more smoothly without losing power.
- Spread wires apart (2s): Giving the wires twice as much breathing room cuts down on capacitance. This keeps them from accidentally storing charge and messing with each other.
- Increase the pitch (2p): Spacing out the centre lines of your wires gives you better isolation. It prevents signals from bleeding into neighbouring lines.
- Use shielding: Think of this as putting up guardrails. Placing extra shield wires next to your most critical lines blocks out unwanted noise and interference.
- Pick the high layers: Always route your most important lines on the upper metal layers. These top layers naturally have lower resistance and capacitance, giving you the cleanest signal path.

6.5 Overcoming Challenges of NDR

Even though Non-Default Routing (NDR) is great for performance, it definitely eats up extra space and can create serious traffic jams during routing. Here is how we usually handle that:

- Use it only where it matters: Don't use NDR everywhere. Save it strictly for your heavy-lifters and critical paths, like clock lines, resets, and high-speed data.
- Drop in buffers: Instead of making a super-long wire wider to fix a delay, break it up by inserting buffers. It saves a lot of space.
- Plan your routing tracks: Take control of track assignment early on. If you guide the signals onto specific tracks strategically, you can prevent layout traffic jams before they start.
- Let the software do the heavy lifting: Lean heavily on your EDA tools. Modern layout tools have great built-in algorithms designed specifically to clean up the mess when NDR complicates things.
- Keep an eye on the big picture: Always ask yourself if the timing boost is actually worth the extra area and power you are burning. It is all about finding that sweet spot.

6.6 Timing Analysis

I wanted to see exactly how custom routing rules (NDR) affect the clock tree. Because the clock network is the backbone of any chip, its performance makes or breaks our timing, skew, and overall reliability.

To optimize it, I broke the clock tree down into three zones and gave each its own rules. First are the top root nets, which carry the main signal across long distances. Next are the trunk nets, which act as the mid-level bridges. Finally, the leaf nets handle the short, last-mile connections that plug directly into the flip-flops.

To figure out how different routing rules affect the clock tree, I ran three separate tests—starting with a standard baseline and then trying out two different tweaks:

1. **The Baseline (2w2s everywhere)**: I started by applying double width and double spacing to both the trunks and the leaves. This gave me a solid benchmark so I had something to compare the other tests against.
2. **The Mixed Setup (Trunk 1.5w2.5s / Leaf 2w2s)**: Next, I decided to narrow the trunk wires a bit but give them extra breathing room, while keeping the leaves thick and wide at 2w2s. I wanted to see how a more relaxed trunk layout would mix with a much stronger leaf layout.
3. **The Balanced Setup (1.5w2.5s everywhere)**: For the last run, I went with a uniform approach and applied the 1.5w2.5s rule across both the trunks and the leaves. The whole goal here was to see if we could save some serious routing space and chip area without hurting our timing performance.

6.6.1 Default NDR (2w2s) Analysis

The experiment carried out utilized the NDR settings at 2w2s, implying that both wire width and wire spacing had been doubled from the original routing rule setting. At the end of the day, the whole goal of this setup was to squeeze as much performance out of our clock tree as possible. By fattening up the wires, we made them much more conductive. At the same time, giving them extra breathing room slashed the coupling capacitance, which kept nasty crosstalk out of the picture.

	ivar(cts,invs_ndr_widths,internal) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m14 0.120'
	ivar(cts,invs_ndr_spacing_multiplier,top_root) 'm5:m14 2'
	ivar(cts,invs_ndr_spacing_multiplier,internal) 'm5:m14 2'
	ivar(cts,invs_ndr_spacing_multiplier,leaf) 'm5:m14 2'
	ivar(cts,invs_ndr_widths,top) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m14 0.120'
	ivar(cts,invs_ndr_widths,leaf) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m14 0.120'
	ivar(cts,invs_ndr_widths,top_root) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m14 0.120'
Default	ivar(cts,invs_ndr_spacing_multiplier,top) 'm5:m14 2'
	ivar(cts,routing,top_root) 'm13-m14'
	ivar(cts,routing_layer_effort,default) 'high'
	ivar(cts,routing,leaf) 'm6-m9'
	ivar(cts,routing,internal) 'm10-m14'
	ivar(cts,routing,top) 'm9-m12'

Fig 6.1: 2w2s NDR Configuration

Of course, we had to find that perfect sweet spot between high performance and total routing chaos. To do that, I strategically divided up the metal layers for different sections of the clock network, which you can see mapped out in Figure 7.1:

- Top root nets: I kept this way up on layers M13 and M14. Because these ultra-thick top layers have almost zero resistance, they give our main clock lines the absolute cleanest path across the chip.
- Internal trunk nets: These handle the mid-level heavy lifting, so I ran them across layers M10 to M14.
- Top nets: I assigned these to the middle-upper tier, specifically layers M9 to M12.
- Leaf nets: Finally, I routed these last-mile connections on layers M6 to M9 as they dive back down to link up directly with the standard cells.

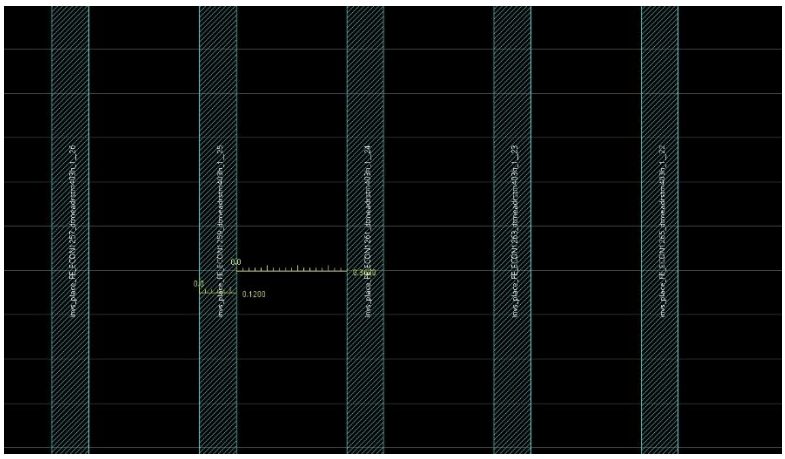

Fig 6.2: Default NDR (2w2s) Layout

Timing Summary:

Setup	mode	all	reg2reg	in2reg	reg2out	in2out	default	bpu	dsbfe	ifu	mem2reg	reg2icg	reg2mem	
WNS	(ns):	-0.204	-0.061	-0.204	-0.084	-0.062	0.261	-0.047	-0.034	-0.04	-0.005	-0.007	-0.034	
TNS	(ns):	-102.924	-1.361	-27.851	-23.248	-0.647	0	-28.095	-1.328	-22.419	-0.04	-0.012	-1.1	-54.355
Violating	Paths:	10626	319	3585	828	24	0	3606	211	2269	25	4	177	
All	Paths:	4.32E+05	1.28E+05	1.11E+05	2251	37	8964	1.97E+05	53769	1.04E+05	209	4305	3559	
	max_nom													
		-0.204	-0.055	-0.204	-0.068	-0.062	0.367	-0.047	-0.034	-0.036	-0.005	-0.004	-0.03	
		-74.598	-1.128	-26.652	-16.586	-0.591	0	-22.496	-1.285	-7.761	-0.014	-0.007	-0.535	-33.226
		8245	243	3328	803	21	0	2839	196	1020	8	2	89	
		4.32E+05	1.28E+05	1.11E+05	2251	37	8964	1.97E+05	53769	1.04E+05	209	4305	3559	
	max_high													
		-0.071	-0.015	-0.071	-0.05	-0.05	0.261	-0.031	-0.018	-0.018	-0.005	0.002	-0.015	
		28.19	-0.173	-11.968	-9.739	-0.199	0	-4.146	-0.075	-2.184	-0.01	0	-0.24	-6.828
		3864	43	2070	699	13	0	783	9	336	8	0	59	
		4.30E+05	1.28E+05	1.11E+05	2251	37	8964	1.95E+05	53769	1.03E+05	209	4305	3475	

Fig 6.3: Snapshot of Innovus Timing Summary (2w2s)

The innovus timing summary showed WNS values around –0.20 ns and TNS exceeding –100 ns, with more than 10,000 violating paths. This acted as the baseline for timing closure. The value written outside the table is the total of reg2reg. For the max_nom case, the reg2reg total is 33.226, which shows the overall timing contribution from register-to-register paths at the nominal corner whereas in max_high case, the reg2reg total is 6.828, reflecting the reduced impact of reg2reg paths under the high corner

Here, in PT (max_nom) the slack values were slightly better, with TNS reduced to about –74 ns and fewer violating paths compared to Innovus.

max_nom					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	-184.751	-42.919	-184.751	-70.401	-55.125
TNS	49331.6	-16235.3	-14722.7	-18141.5	-232.112
NUM	4917	2331	1733	839	14

max_high					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	-182.31	-27.774	-182.31	-52.505	-40.577
TNS	-40595.7	-5152.46	-19641.2	-15610.2	-191.815
NUM	4686	915	2756	1001	14

Fig 6.4: Snapshot of PrimeTime Timing Summary (2w2s)

In (max_high), the results improved further, with WNS around –0.07 ns and a significant reduction in violations (about 3,800 paths), confirming stronger closure at this corner.

6.6.2 Trunk 1.5w2.5s and Leaf 2w2s Analysis

In this case, I have employed the use of trunk 1.5w2.5s and leaf 2w2s. Trunk nets were laid out using the 1.5 width and 2.5 spacing, while the leaf nets kept the same 2w2s. In this case, the

intention was to see what happens when there is a wider, more spaced trunk in combination with stronger leaf nets. In this case, the expectation was that widening and spacing the trunk would reduce resistance and coupling, thereby improving the timing margins. However, by keeping the leaf nets intact, the 2w2s ensured that the local connections were strong.

```
ivar(cts,invs_ndr_widths,internal) 'm6 0.06 m7 0.06 m8 0.06 m9 0.06 m10 0.06 m11 0.06 m12 0.06 m13 0.09 m14 0.09 m15 0.12 m16 0.12 '
ivar(cts,invs_ndr_spacing_multiplier,top_root) 'm5:m16 2'
ivar(cts,invs_ndr_spacing_multiplier,internal) 'm6:m14 2.5'
ivar(cts,invs_ndr_spacing_multiplier,leaf) 'm5:m14 2'
ivar(cts,invs_ndr_widths,top) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m120.080 m13 0.120 m140.120'
ivar(cts,invs_ndr_widths,leaf) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m120.080 m13 0.120 m140.120'
ivar(cts,invs_ndr_widths,top_root) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m140.120'
ivar(cts,invs_ndr_spacing_multiplier,top) 'm5:m14 2'

ivar(cts,routing,top_root) 'm15-m16'
ivar(cts,routing_layer_effort,default) 'high'
ivar(cts,routing,leaf) 'm6-m11'
ivar(cts,routing,internal) 'm10-m16'
ivar(cts,routing,top) 'm9-m12'
```

Fig 6.5: Trunk 1.5w2.5s and Leaf 2w2s Configuration

Routing was carried out by assigning the root nets to M15-M16 layers, the internal trunk nets were assigned to M10-M16 layers, the top nets were routed using M9-M12 layers, and the leaf nets were kept to M6-M11 layers. The layer assignment helped ensure that the reinforced trunk net used the low-resistance upper layers, while leaf nets were kept in mid-level layers.

Timing Summary:

Setup	mode	all	reg2reg	in2reg	reg2out	in2out	default	bpu	dsbfe	ifu	mem2reg	reg2icg	reg2mem	
WNS	(ns):	-0.179	-0.071	-0.179	0.08	-0.071	0.271	-0.044	-0.04	-0.045	-0.01	-0.01	-0.032	
TNS	(ns):	-98.591	-1.64	-17.921	-25	-0.596	0	-34.868	-1.526	-19.248	-0.095	0.038	-2.011	-59.35
Violating	Paths:	11343	424	340	919	24	0	422	351	2281	28	9	312	
All	Paths:	4.33E+05	1.28E+05	1.11E+05	225	37	9095	1.97E+05	53751	1.04E+05	209	4289	3559	
	max_nom													
		0.179	-0.071	-0.179	-0.066	-0.071	0.375	-0.041	-0.04	-0.036	-0.01	-0.01	-0.026	
		-71.956	-1.245	-16.057	-19.854	-0.583	0	-27.141	-1.443	-8.216	-0.048	-0.018	-1.158	-39.269
		8568	269	2942	891	22	0	3316	326	1099	17	3	138	
		4.33E+05	1.28E+05	1.11E+05	2251	37	9095	1.97E+05	53751	1.04E+05	209	4289	3559	
	max_high													
		-0.058	-0.019	-0.058	0.04	-0.029	0.271	-0.033	-0.025	0.016	-0.004	-0.002	0.018	
		-26.583	-0.3	-7,511	-10.873	-0.148	0	-6.494	-0.158	-1.183	-0.009	-0.005	-0.982	-9.131
		345	95	1280	783	13	0	1050	29	282	6	3	134	
		4.30E+05	1.28E+05	1.11E+05	225	37	9095	1.95E+05	53751	1.03E+05	209	4288	3475	

Fig 6.6: Snapshot of Innovus Timing Summary (Trunk 1.5w2.5s and Leaf 2w2s)

The Innovus time summary had a WNS of about –0.18 ns and a TNS that was almost –98 ns with more than 11,000 violating paths. In the max_nom case, the reg2reg total stands at 39.269, which is the overall contribution made by reg2reg paths under the nominal corner.

The reg2reg total in the max_high case stands at 9.131, indicating lower impact made by reg2reg paths under the high corner.

After moving to PrimeTime in the nominal corner (max_nom), there were improvements in the slack values. The WNS was still negative but better compared to Innovus, while the TNS was lowered to –72 ns.

max_nom					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	169.851	55.176	-169.851	-68.67	-36.868
TNS	52930.9	-26062.7	6518.69	-20217.5	-132.019
NUM	5316	346	924	917	11

max_high					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	168.293	31.377	168.293	48.038	-27.265
TNS	35485.1	-7844.02	-11505	-16018.2	-117.882
NUM	4690	134	2297	1039	11

Fig 6.7: Snapshot of PrimeTime Timing Summary (Trunk 1.5w2.5s and Leaf 2w2s)

At the high corner (max_high), the results improved further. The WNS was around –0.05 ns and the TNS dropped to about –26 ns. The violating paths reduced to nearly 3,400, showing that timing closure was much stronger under this condition.

6.6.3 Trunk 1.5w2.5s and Leaf 1.5w2.5s Analysis

In this, I have applied trunk 1.5w2.5s and leaf 1.5w2.5s. Both trunk and leaf nets were given 1.5width along with 2.5spacing, so the configuration was uniform across the clock tree. The idea was to strengthen every level of the network instead of only focusing on trunk or leaf separately.

The top root nets were allocated to M15-M16 levels, internal nets were distributed from M9 to M16 levels, top nets were allocated to M9-M12 level, and leaf nets were allocated to M6-M11 levels. In such a way, the design was allocated using high metal layers for trunk and mid-level metal layers for leaf.

```
ivar(cts,invs_ndr_widths,internal) 'm6 0.06 m7 0.06 m8 0.06 m9 0.06 m10 0.06 m11 0.06 m12 0.06 m13 0.09 m14 0.09 m15 0.12 m16 0.12 '
ivar(cts,invs_ndr_spacing_multiplier,top_root) 'm5:m16 2'
ivar(cts,invs_ndr_spacing_multiplier,internal) 'm6:m14 2.5'
ivar(cts,invs_ndr_spacing_multiplier,leaf) 'm5:m14 2.5'
ivar(cts,invs_ndr_widths,top) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m120.080 m13 0.120 m140.120'
ivar(cts,invs_ndr_widths,leaf) ' m6 0.06 m7 0.06 m8 0.06 m9 0.06 m10 0.06 m11 0.06 m12 0.06 m13 0.09 m14 0.09 m15 0.12 m16 0.12'
ivar(cts,invs_ndr_widths,top_root) 'm5 0.048 m6 0.080 m7 0.080 m8 0.080 m9 0.080 m10 0.080 m11 0.080 m12 0.080 m13 0.120 m140.120'
ivar(cts,invs_ndr_spacing_multiplier,top) 'm5:m14 2'

ivar(cts,routing,top_root) 'm15-m16'
ivar(cts,routing_layer_effort,default) 'high'
ivar(cts,routing,leaf) 'm6-m11'
ivar(cts,routing,internal) 'm9-m16'
ivar(cts,routing,top) 'm9-m12'
```

Fig 6.8: Trunk 1.5w2.5s and Leaf 1.5w2.5s Configuration

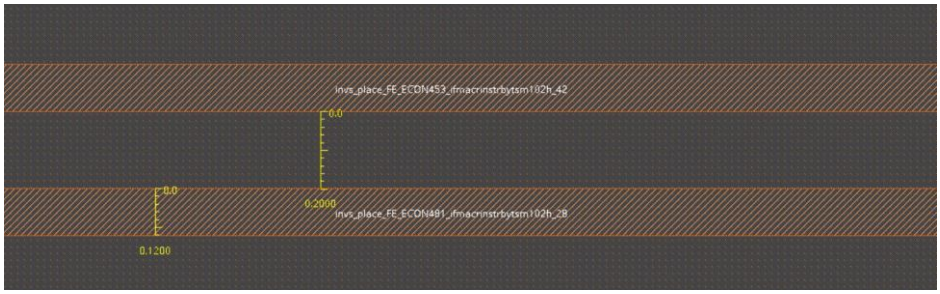

Fig 6.9: 1.5w and 2.5s NDR Layout

Timing Summary:

Setup	mode	all	reg2reg	in2reg	reg2out	in2out	default	bpu	dsbfe	ifu	mem2reg	reg2icg	reg2mem	
WNS	(ns):	-0.181	-0.083	-0.181	-0.079	-0.082	0.27	-0.043	-0.029	-0.049	-0.015	-0.004	-0.026	
TNS	(ns):	-115.49	-1.977	-32.195	-22.571	-0.719	0	30.855	-1.14	-31.219	-0.177	-0.015	-1.123	-4.796
Violating	Paths:	11469	482	4168	828	22	0	3710	238	2478	39	8	19	
All	Paths:	4.34E+05	1.28E+05	1.12E+05	2251	37	9488	1.97E+05	53756	1.04E+05	209	436	3559	
	max_nom													
		-0.181	-0.083	-0.181	-0.062	-0.082	0.37	-0.039	-0.029	-0.036	-0.009	0	-0.019	
		-79.002	-1.277	-30.016	-15.434	-0.699	0	-23.454	-1.064	-11.743	-0.036	0	-0.489	-38.063
		8237	276	3670	797	17	0	2825	219	903	8	0	64	
		4.34E+05	1.28E+05	1.12E+05	2251	37	9488	1.97E+05	53756	1.04E+05	209	436	3559	
	max_high													
		-0.058	-0.028	-0.058	-0.034	-0.033	0.271	-0.03	-0.02	-0.016	0.002	-0.001	0.012	
		-31.556	-0.397	-15.041	-8.411	-0.155	0	-6.25	-0.091	-2.072	-0.006	-0.002	-0.295	-9.113
		4287	93	2241	677	13	0	984	1	448	5	3	68	
		4.31E+05	1.28E+05	1.12E+05	2251	37	948	1.95E+05	53756	1.03E+05	209	436	3475	

Fig 6.10: Snapshot of Innovus Timing Summary (Trunk 1.5w2.5s and Leaf 1.5w2.5s)

For the Innovus timing analysis, the worst negative slack was approximately –0.18 ns, whereas the total negative slack was –115 ns, and the number of paths violating the constraint exceeded 11,000 paths. Under nominal operating conditions (max_nom), the register-to-register paths exhibited an aggregate timing contribution of 38.063. This parameter was isolated outside the primary tabular data to distinctly highlight its magnitude and influence on the overall design constraints. Conversely, shifting the analysis to the max_high corner resulted in a pronounced reduction in the register-to-register total, which fell to 9.113.

The timing profile exhibited noticeable improvement upon transferring the nominal dataset to the sign-off environment in PrimeTime. Although the worst negative slack (WNS) remained negative, the severity of the violation was significantly mitigated relative to the initial electronic design automation (EDA) tool estimates. Furthermore, the total negative slack (TNS) contracted to –79 ns, accompanied by a reduction in the total number of violating paths to approximately 8,200.

max_nom					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	-165.793	-75.419	-165.793	-68.838	-50.654
TNS	-55246.6	-20967.5	-17370.8	-16677.2	-231.143
NUM	5264	2662	1764	828	10

max_high					
	Total	reg->reg	in->reg	reg->out	in->out
WNS	-166.461	-30.124	-166.461	-42.606	-36.41
TNS	-46263.6	-9019.37	-22708.9	-14388.8	-146.569
NUM	5821	1523	327	1009	11

Fig 6.11: Snapshot of PrimeTime Timing Summary (Trunk 1.5w2.5s and Leaf 1.5w2.5s)

By proceeding further with PrimeTime at max_high corner, the results were even better. In this case, WNS was near to –0.05 ns and TNS reduced to –31 ns. The number of violating paths was approximately 4,300 in this scenario.

6.7 Conclusion Summary

The comparison among the three NDR types is clear with respect to clock ID and timing values. The default type, Trunk 2w2s and Leaf 2w2s, gives a clock ID of 110.2 and was used as a base. With the trunk changed to 1.5w2.5s and the leaf to 2w2s, the clock ID was reduced to 109.6, resulting in a slight improvement. Lastly, when both the trunk and leaf are the same (1.5w2.5s), this yielded a lower clock ID of 104.8, the best result among all experiments.

The comparison of Innovus and PrimeTime timing reports also follows the same pattern: Innovus gave the baseline with more violations, and PrimeTime nominal and high corners gave smaller TNS with fewer violation paths. The 1.5w2.5s combination for both tools gave the best closure and the smallest clock ID, proving to be the best NDR to use.

As a result, the Trunk 1.5w2.5s, Leaf 1.5w2.5s gave the best timing and clock ID improvements

CHAPTER 7

TIMING FIXING WITH ECO

7.1 Introduction to ECO

Engineering Change Order (ECO) is an essential part of the VLSI design flow which enables changes to be carried out on the design of a chip post synthesis and placement phase. In particular, ECO timing fix refers to specific fixes that are carried out in order to fix problems relating to timing detected using static timing analysis (STA) without requiring any re-design or re-implementation of the whole chip. This is usually done towards the latter phases of design and in a way that does not disrupt the whole design process.

7.2 Architectural Context within the Physical Design Flow:

In the conventional PD implementation flow, the design goes through certain predictable and constructive stages in a linear fashion, including:

- Synthesis
- Netlist Generation
- Floorplanning
- Power Grid Synthesis
- Placement and Optimization
- Clock Tree Synthesis (CTS)
- Routing.

Once Routing is completed, the geometric information of all signal paths is fully defined in the virtual grid. However, as the P&R router employs simplified delay models to estimate delays inside itself, the design needs to go through the Sign-off & ECO Loop. This is a standalone verification pipeline aimed at fixing any problems in the design without affecting its well-established parts.

7.3 Comprehensive Breakdown of the Iterative ECO Cycle:

Implementation of the ECO process requires a strict four-step loop that connects the physical reality of the physics to logical intention. Every step must be implemented in sequence where specific data types are provided to the next tool engine.

Stage 1: Sign-off Parasitic Extraction (RC Extraction)

The main aim of this step is to get precise electrical characteristics of routed metal interconnects. The geometric layout data in the form of DEF or GDS format is examined using independent high precision Sign-off Extraction tool such as Synopsys StarRC or Cadence QRC. The tool measures the precise cross-sectional area, material resistance (R), and multi-body capacitance (C) of every copper or aluminium wire trace, including vertical interlayer vias and lateral wire-to-wire spacing parameters. At sub-nanometre nodes, the extraction engine cannot model wires as simple, single capacitors. Instead, it extracts complex distributed networks tracking coupling capacitance, intrinsic/ground capacitance, and via resistance. Coupling capacitance isolates the capacitance which is critical for predicting crosstalk delay and crosstalk glitches.

Stage 2: Multi-Dimensional Parallel Sign-off Engines

The generated SPEF file and the current gate-level netlist are verified simultaneously across three distinct criteria using independent tool engines to isolate any layout failures.

A. Static Timing Analysis (STA)

STA sign-off engines like Synopsys' PrimeTime or Cadence's Tempus analyze SPEF data to compute gate propagation delays and signal transition delays accurately. The analysis is done over multiple PVT (process, voltage, temperature) corners in order to detect two kinds of violations with the help of stringent mathematical slack equations. The setup slack is defined as the difference between required and arrival times. Setup violations happen when the data requires a lot of time to traverse a physical path because of high wire resistances or excessive capacitive loadings that cause the signal to be captured after the edge window of the clock. On the other hand, hold slack is the difference between the arrival time and required time. Hold violations happen when the data travels very quickly along short or heavily loaded paths and the signal overwrites the stable state of the data before the current clock edge captures it. Tools first operate in GBA mode to detect potential violations in an efficient but pessimistic way. Violating paths are analyzed again in PBA mode to compute exact physics of the violating paths.

B. Logic Equivalence Checking (LEC)

If any bugs are found in the source code later in the process, the front-end engineering team will update the RTL. The LEC tool (e.g., Cadence Conformal/Synopsys Formality) is used to

compare this updated golden RTL description with the current post-routed gate netlist, identifying the logic differences that must be modified on the floorplan.

C. Physical Verification (DRC/LVS)

Physical rule checkers, such as Siemens Caliber, use strict foundry design rules to verify geometric patterns. Design Rule Check verifies that there are no violations related to spacing and width of metal lines, as this would cause a physical short or break in production. LVS checks that the geometric patterns match the netlist connections.

D. The Sign-off Decision Matrix:

After the parallel execution of the sign-off engines is complete, the metrics are forwarded to a central evaluation point. The tool checks whether any violations regarding timing, physical, or logical exist. If Slack is found to be positive or zero in setup and hold margins and no DRC, LVS, or LEC errors are observed, the design would exit the process directly without involving the ECO infrastructure and become ready for tape-out. Any violation would result in choosing the "Yes" path, leading to Stage III.

Stage 3: Algorithmic ECO Script Generation

Whenever a sign-off engine encounters an error, the automated ECO engine analyzes the design locally and computes the least invasive fix to address the problem. Instead of creating a new netlist file, the software generates a Tcl (Tool Command Language) script to modify specific instances.

The setup problems are fixed using several algorithms that manipulate three main settings. First, it employs gate sizing, in which the software can replace a weak cell with a stronger one to discharge the subsequent RC network faster. It verifies whether the additional input capacitance of the gate increases the delay of the previous stage. Second, because the wire delay scales quadratically with length, it separates very long wires by adding buffers, thereby converting a long RC network into a linear one. Finally, threshold voltage swapping is achieved by replacing a slow cell with an equally sized but faster one.

Because hold violations cannot be fixed by lowering operating frequencies on a manufactured chip, the engine treats them with high priority. It inserts highly stable, specialized delay buffers right before the capturing sequential register. The algorithm must insert sufficient delay to clear

the hold violation at the fastest corner (Best-Case) without introducing a setup violation at the slowest corner (Worst-Case).

Stage 4: Incremental Physical Implementation

The Tcl patch script is loaded directly back into the core Place and Route (P&R) layout engine, feeding into the pipeline right at the Routing implementation checkpoint. The tool automatically resolves cell overlap caused by newly added components. Then it performs an isolated ECO routing process to ensure that the new connections integrate properly with the layout database. During placement legalization, added or expanded cells must be placed legally within standard row heights without overlapping neighboring circuits. The tool identifies localized empty filler slots on the floor plan and drops the cell into them. If the area is full, the tool executes a highly localized kinematic push, nudging adjacent non-critical cells or decoupling capacitors over by a few routing pitches to clear out an exact slot for the new ECO cell. During incremental routing, the tool utilizes an ECO-routing mode. It un-routes and deletes only the specific signal wires tied to the modified pins, leaving the rest of the chip's routing grids completely locked. It then weaves new localized metal wires through the available routing channels to complete the patch. For severe setup violations, the tool may apply layer swapping, moving the net from lower, highly resistive metal layers to thick, top-level ultra-low-resistance global metal layers.

7.4 The Loop Closure Mechanism:

Once incremental routing is complete, the layout's physical geometry has changed. Because physical features on an advanced node chip are tightly coupled, the layout must re-enter the verification pipeline to ensure that the modifications have successfully cleared the violations and have not disrupted neighboring passing networks. This new layout geometry is then entered directly into the Sign-off Parasitic Extraction tool to create a new SPEF file. It is then analyzed using the parallel Static Timing Analysis, Logic Equivalence Checking, and Design Rule Checking tools. This process is then repeated iteratively by the designers and the automation scripts until the unified iteration results in a clean sheet with no timing violations, perfect logical equivalence, and no DRC violations. At this stage, the finalized layout is then exported to a GDSII/OASIS database file format and safely delivered to the foundry for processing (Tape-out).

7.5 Detecting Timing Issues:

ECO starts by performing Static Timing Analysis (STA) to identify setup and hold time violations in the design. This means the signals do not reach the flip-flop input ports in time to capture valid data during the active clock period. This might be due to high combinational path delays, clock skew, or poor placement and routing. If such faults are not corrected, they might lead to data corruption, malfunctions, or system unreliability in the long run. Only through accurate detection of timing problems using STA are problem paths identified and ECO solutions implemented.

7.6 Evaluating Timing Issues:

Every timing violation found is thoroughly analyzed to identify the cause and take appropriate action. Setup timing violations usually occur due to large data path delays, which prevent the signal from arriving at the flip-flop input within the specified setup time period. Hold timing violations, on the other hand, are caused by a small data path delay, which results in data changing too soon after the clock transition. It is critical to know whether the problem is due to too much or too little delay, as this determines the appropriate ECO implementation method.

7.7 Formulating ECO Issues:

A variety of tactics are then formulated to address the timing issues, including:

- **Adjusting Gate Sizes:**

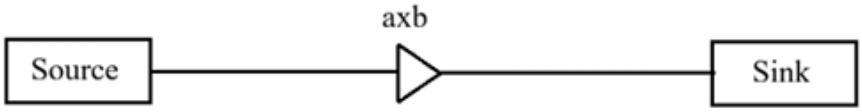

Fig 7.1: Original Buffer Sizing before ECO

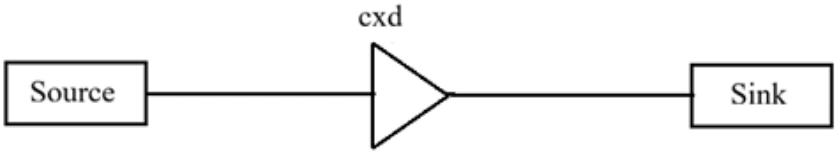

Fig 7.2: Buffer Upsizing after ECO for Timing Improvement

The figure above illustrates a timing ECO technique based on gate resizing, where the drive strength of a logic cell is modified to correct timing violations. In the figure 13, the buffer with drive strength axb has a lower drive capacity, resulting in higher delay along the data path and potentially causing a setup timing violation at the sink. In ECO, this buffer gets replaced by the higher-drive strength cell cxd. It gives better output drive capability and less propagation delay. The gate size being increased helps in faster transition of signals, thus allowing the data to arrive at the destination during the time window as depicted in Figure 14. The selected resizing of gates is an effective way to improve timing without changing the circuit's functionality.

- **Buffer Management:**

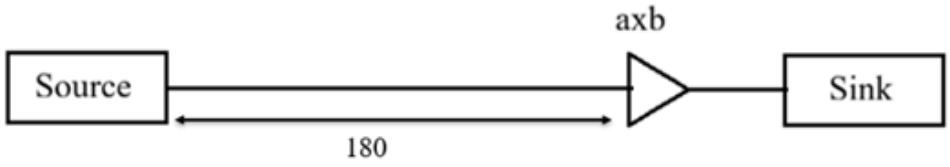

Fig 7.3: Single-Buffer Path with Long Signal Propagation Distance

Buffer management is a widely used ECO technique to correct timing violations caused by long or short interconnect delays. In designs with long signal propagation distances, excessive RC (resistance–capacitance) delay can lead to setup timing violations, in which data arrives late at the sink, as shown in the figure above (15). To mitigate this, buffers are inserted along the interconnect—often at the midpoint—to divide the long wire into shorter segments, thereby reducing delay and improving signal transition quality.

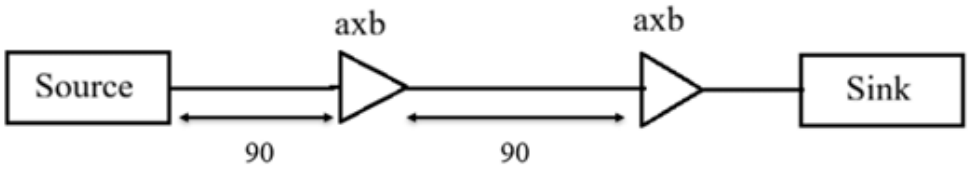

Fig 7.4: Buffer Insertion with Equal Driving Strength for Distance Partitioning

To overcome setup timing violations caused by delays introduced by long interconnects, buffers are inserted between these paths, usually midway along the interconnect. The long interconnects have high resistance and capacitance, resulting in high delay and poor transition times. With buffering, the long interconnect can be broken into smaller interconnect segments, reducing RC delay and improving the speed of signal transition through it. Hence, by buffering,

the signal's drive strength is improved, the path delay is minimized, and the signal reaches the receiver within the allotted setup timing window.

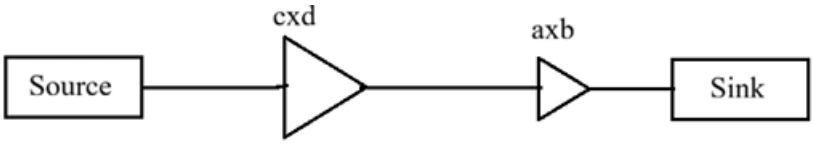

Fig 7.5: Buffer Insertion Using Different Driving Strengths for Timing Optimization

Sometimes, strong buffers are added to increase the speed of data propagation along critical paths, because they can handle the increased capacitance, reduce delay, and thus help overcome setup violations. However, excessive buffering or a too-strong buffer may cause data to propagate too quickly, so the data reaches the receiver flip-flop before the specified time. Here, timing is fixed by reducing unnecessary buffer usage, decreasing the buffer strength, or increasing the delay path, thereby satisfying hold requirements while retaining functionality.

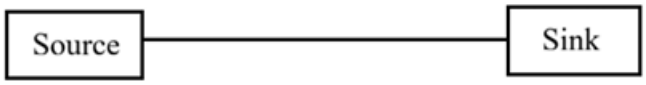

Fig 7.6: Direct Source-to-Sink Connection after Buffer Removal

In case of violations that take place due to hold time, the elimination of unrequired buffers or replacement of the same by buffers having reduced drive strength helps in increasing the delay of the data path in a way that signals do not reach the flip-flop receiving the signals before time. In this way, the circuit is made to meet the criteria of hold time without affecting the functionality of the circuit.

- **Logic Duplication:**

Logic duplication is a general technique in VLSI design that is used to reduce timing delays caused by high fanout.

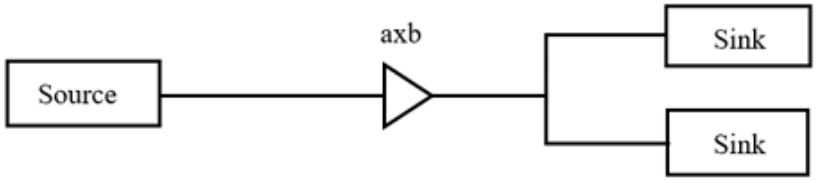

Fig 7.7: Single Buffer Driving Multiple Fanouts

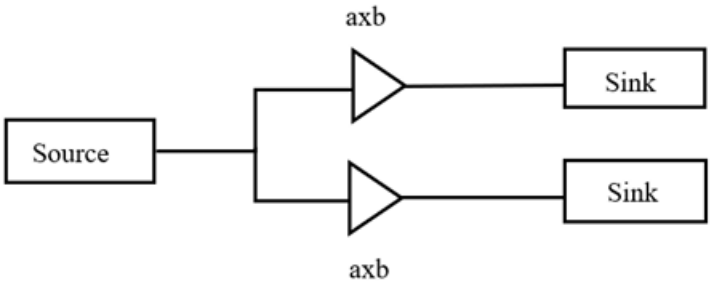

Fig 7.8: Logic Duplication with Separate Buffers for Each Fanout

If one buffer feeds into several sinks, it will become overloaded with all the load coming from them. This will slow the signal due to higher capacitance. It would make sense to replicate the buffer so that it could feed into different regions of the chip. In this way, each buffer will have to handle a lower load, and the signal will travel faster. It is possible to optimize delays, improve timing, and even reduce power consumption by sharing the load among multiple buffers. Replicating the logic and distributing it across the chip can speed up signal delivery.

- **Netlist Adjustments:**

The ECO netlist modification process in chip design can be compared to making minor corrections to the existing circuit design rather than creating a new one. While the whole process of chip design usually begins from the synthesis of RTL description into a gate-level netlist, the ECO process targets only those elements of the netlist that require correction. Usually, resizing certain gates, adding additional buffers, and other modifications are needed to satisfy timing requirements. Such modifications are made using ECO scripts that incrementally change the existing design database. It means that most of the verified netlist remains untouched, and only specific portions are modified as required.

7.8 Applying ECO Adjustments:

When using the ECO strategy, all manipulations should be performed accurately and thoughtfully. The manipulations target specific areas within the design – either the netlist, the layout, or even the clock tree. The point is to solve some timing or functionality problems, but not to bring any additional problems as a result of the performed changes. Everything being done should be considered carefully, since the solution to the current problem cannot cause other negative consequences, such as increased chip area, power dissipation, or routing complexity.

7.9 Validating ECO Adjustments:

The design passes a full validation steps following ECOs to ensure the time issues have been finished. This is mainly done by repeating the static timing analysis (STA) to check whether necessary timing paths fulfils the requirements specified. Additional power and signal integrity validations are required. STA checks whether the data has been obtained correctly by flipflops in a synchronous circuit considering both setup and hold times.

Positive hold slack means the data will be stable long enough to meet the hold requirement; n egative hold slackreflects a timing violation that should be resolved to verify that the design functions as desired.

Static Timing Analysis (STA): STA is a procedure used to ensure that a flip-flop successfully captures data in a synchronous digital design.

Setup Time Violation Test: The setup time test verifies that the data arrives at the flip-flop's input before the clock edge. The setup margin, also referred to as setup slack, is determined by the difference between Trequired and Tarrival. Set up Slack Formula:

Setup Slack = T_required - T_arrival(max)

Hold Violation Test: In the hold time test, it ensures that the data is stable enough after the clock edge to be captured correctly by the flip-flop. Hold slack or hold margin is defined as the difference between the time at which the data arrives (Tarrival) and the time at which data should have arrived and become stable (Trequired).

The hold slack is positive when the data is stable for the hold time requirement, thus satisfying the hold constraint. Negative hold slack represents the hold violation case when data becomes unstable before the end of hold time. In both cases, the negative slack represents the violation of the timing requirement and needs to be addressed.

7.10 Engineering Best Practices

1. **Netlist Freeze Stability:** Maintain strict netlist freeze boundaries across engineering sub-teams before generating ECO scripts to avoid chasing moving targets within the layout database.
2. **Crosstalk SI Awareness:** Always utilize sign-off tools that incorporate Signal Integrity (SI) options during ECO generation to prevent new wire routes from introducing catastrophic noise or crosstalk glitching on adjacent lines.

3. **Fix Hold Over Setup:** When resolving concurrent violations under tight schedules, prioritize hold corrections over setup corrections; a minor setup violation can often be bypassed in final test by reducing external operating frequency, whereas a hold violation causes fundamental silicon failure that cannot be recovered.

CHAPTER 8

RESULTS

8.1 RESULTS

This chapter highlights the results from the project implementation, particularly timing optimization to guarantee proper setup and hold time specifications. This objective has been accomplished through the use of a customized Place & Route (P&R) tool whose functionality and efficiency have been improved with time.

- **Setup Results:**

Pre ECO Results:

Table 2: Setup Timings before ECO

	TOTAL	REG-REG	IN-REG	REG-OUT	IN-OUT
WNS	-348.749	-72.768	-214.899	-348.749	-289.11
TNS	-37223.14	-8613.04	-22923.893	-348.749	-2647.767
NUM	1739	979	327	419	14

Post ECO Results:

Table 3: Setup Timings after ECO

	TOTAL	REG-REG	IN-REG	REG-OUT	IN-OUT
WNS	-158.978	-56.32	-124.753	-246.97	-231.98
TNS	-18652.23	-3675.907	-15824.87	-298.07	-2365.67
NUM	956	453	236	342	10

Setup timing violations were noticed in the design before the application of the ECO, which meant that some of the signals could not be propagated within the required target clock period. With the application of the ECO, all these setup violations have been solved, and the design now satisfies all the required timing requirements. The ECO process has helped to achieve

optimal timing in the design, thus guaranteeing the proper functionality of the design at the desired clock frequency.

- **Hold Results:**

Pre ECO Results:

Table 4: Hold Timing before ECO

	TOTAL	REG-REG	IN-REG	REG-OUT	IN-OUT
WNS	-648.739	-123.98	-145.89	-350.56	-301.54
TNS	-45323.23	-3568.09	-17894.72	-357.90	-2978.45
NUM	2354	850	273	289	54

Post ECO Results:

Table 5: Hold Timing after ECO

	TOTAL	REG-REG	IN-REG	REG-OUT	IN-OUT
WNS	-534.89	-98.80	-105.45	-123.56	-256.31
TNS	-34569.09	-2980.78	-10523.89	-234.35	-2100.78
NUM	1989	750	145	170	32

In the absence of the ECO, there were hold timing violations on some paths, meaning that the arrival of signals was too early, thus resulting in wrong data capture. However, once the ECO was applied, the violations were corrected, and the paths have met the hold timing criteria. The ECO helped solve the problem of the early arrival of signals to ensure that the transfer of data between registers occurred successfully.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

This project demonstrates the full physical design flow of an advanced X86 processor by creating a GDSII-ready layout from synthesized RTL. This project involved efficient placement and floor-planning methods that timed convergence in both internal and interface paths. Beginning with the basics of physical design, all the necessary concepts of automated place and route (APR) have been covered up to the point where real-time problems were solved. All sorts of partitions were made, and corresponding problems had to be solved using different methods. Caliber reports were analyzed, and various violations, such as max transition and max output capacitance, were identified and corrected. Both clock and data path timing violations were addressed through iterations until all were resolved. There were many rounds of ECO to address the remaining problems and implement certain functional changes. Moreover, timing-driven NDR analysis was performed, which provided insight into the effect of non-default routing on clock distribution and timing closure. Through experimentation with different trunk and leaf combinations, it was shown that NDR is a very powerful optimization tool. This research paper presents a case study of the successful application of a systematic physical design methodology to achieve timing closure, power optimization, and manufacturability criteria. It serves as a good basis for further advancements in chip design.

9.2 FUTURE SCOPE

The future scope of this research involves applying physical design approaches to address the needs of the emerging generation of semiconductor technology. Advanced nodes, chiplet-based design approaches, and 3D IC integration have become requirements for design flows that need to accommodate increasing complexity, manufacturability, and heterogeneous integration. Use of machine learning in the placement, routing, and time-closure processes will enable faster convergence and make predictions of issues such as congestion and power more accurate. Power intent propagation using UPF may be used for dynamic voltage scaling and better control of power gates to create ultralow-power designs. Lastly, cross-domain co-optimization

that integrates hardware and software design will create numerous possibilities for the development of AI accelerators, high-performance computers, and edge devices. Taking into account the ideas described in Chapter 6, my next step will be to explore more advanced NDR techniques based on a timing-driven approach. In particular, testing multilayer architectures and parameter optimization might yield even greater benefits for clock distribution and timing closure.

REFERENCES

[1] Zhou, Yanling, Yunyao Yan, and Wei Yan. "A method to speed up VLSI hierarchical physical design in floorplanning." *2017 IEEE 12th International Conference on ASIC (ASICON)*. IEEE, 2017.

[2] Kumar, Gundrathi Vinod, et al. "Optimizing Timing and Physical DRC Performance in VLSI Design: A Comprehensive Approach." *2024 5th IEEE Global Conference for Advancement in Technology (GCAT)*. IEEE, 2024.

[3] Bharath, Duttuluru, A. Jagadeesh, and T. Jaya. "Analysis of CAD tools in VLSI Physical design." *2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2025.

[4] Gowda, Mahendra Shivaram, Chambamma Koti, and Poornima Mohanachandran. "Power optimization techniques during synthesis and physical design for a low-power RISC-V design." *2024 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*. IEEE, 2024.

[5] Kulkarni, Atharva M., and Abhay Chopde. "Physical design: Methodologies and developments." *arXiv preprint arXiv:2409.04726* (2024).

[6] Darmi, Mohammed, et al. "Integrated circuit conception: A wire optimization technic reducing interconnection delay in advanced technology nodes." *Electronics* 6.4 (2017): 78.

[7] Vishnu, Priya V., A. R. Priyarenjini, and Naveen Kotha. "Clock tree synthesis techniques for optimal power and timing convergence in soc partitions." *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. IEEE, 2019.

[8] Maestre, Susie, et al. "Improving digital design PPA (performance, power, area) using iSpatial physical restructuring." *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. IEEE, 2022.

[9] Weng, Wanzheng, and Pingqiang Zhou. "RapidPnR: Accelerating the physical design for FPGAs via design-level parallelism." *Integration* (2025): 102532.

[10] Chen, Jianli, and Wenxing Zhu. "An analytical placer for VLSI standard cell placement." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.8 (2012): 1208-1221.

[11] Huang, Ying-Yao, et al. "Learning based placement refinement to reduce DRC short violations." *2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, 2021.

[12] Adya, Saurabh N., and Igor L. Markov. "Fixed-outline floorplanning: Enabling hierarchical design." *IEEE transactions on very large scale integration (VLSI) systems* 11.6 (2003):1120-1135.

[13] Otten, Ralph HJM. "Automatic floorplan design." *19th Design Automation Conference*. IEEE, 1982.

[14] Li, Li, et al. "PS-FPG: Pattern selection based co-design of floorplan and Power/Ground network with wiring resource optimization." *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2010.

[15] AshokKumar, N., K. Nagendra, and P. ShanthaKumar. "Enhanced Floor plan Algorithm for Ultra Complex SoC's."

[16] Singh, Sonipriya, et al. "Pessimism reduction in voltage-aware drc using simulation results for functionally correlated nets." *2023 IEEE Women in Technology Conference (WINTECHCON)*. IEEE, 2023.

[17] Tseng, Tsu-Wei, et al. "A power delivery network (PDN) engineering change order (ECO) approach for repairing IR-drop failures after the routing stage." *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*. IEEE, 2014.

[18] Vishnu, Priya V., A. R. Priyarenjini, and Naveen Kotha. "Clock tree synthesis techniques for optimal power and timing convergence in soc partitions." *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. IEEE, 2019.

[19] Hashimoto, Masanori, Masao Takahashi, and Hidetoshi Onodera. "Crosstalk noise optimization by post-layout transistor sizing." *Proceedings of the 2002 international symposium on Physical design*. 2002.

PLAGIARISM REPORT

DrillBit

DrillBit Similarity Report

7	85	A	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	pdfcookie.com	<1	Internet Data
2	vlsitesting.wordpress.com	<1	Publication
4	dl.lib.uom.lk	<1	Internet Data
5	eced.thapar.edu	<1	Internet Data
6	geeksforgeeks.org	<1	Internet Data
7	pdfcookie.com	<1	Internet Data
8	sjec.ac.in	<1	Publication
10	Student Thesis Published in HAL Archives	<1	Publication
11	Thesis Submitted to Shodhganga Repository	<1	Publication
12	nics.ee.tsinghua.edu.cn	<1	Internet Data
13	worldwidescience.org	<1	Internet Data
14	www.ascdegreecollege.ac.in	<1	Publication
15	moam.info	<1	Internet Data
16	scholarworks.umass.edu	<1	Publication

Dr. Gitanjali Chandwani Manocha
Assistant Professor,
Department of ECE,
Thapar Institute of Engineering & Technology, Patiala

Dr. Hem Dutt Joshi
Professor & Associate Dean
(DCT), Department of ECE,
Thapar Institute of Engineering & Technology, Patiala